REAL-TIME FLIGHT PATH OPTIMIZATION FOR TRACKING STOP-AND-GO TARGETS WITH MICRO AIR VEHICLES

THESIS

Nate A. Terning, Captain, USAF
AFIT/GAE/ENY/08-M28

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/ENY/08-M28

REAL-TIME FLIGHT PATH OPTIMIZATION FOR TRACKING STOP-AND-GO TARGETS WITH MICRO AIR VEHICLES

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Nate A. Terning, BS

Captain, USAF

March 2008

AFIT/GAE/ENY/08-M28

REAL-TIME FLIGHT PATH OPTIMIZATION FOR TRACKING STOP-AND-GO TARGETS WITH MICRO AIR VEHICLES

Nate A. Terning, BS
Captain, USAF

Approved:

/signed/

_____ _____
Dr. Richard G. Cobb (Thesis Advisor) date

/signed/

_____ _____
Major Paul A. Blue (Member) date

/signed/

_____ _____
Dr. David R. Jacques (Member) date

Abstract

This research effort focuses on using a heuristic approach to determine the optimal flight path required to put an Unmanned Aircraft System's (UAS) sensor on a moving target in the presence of a constant wind field. This thesis builds on past work using dynamic optimization techniques to calculate minimum time to target. The computationally intensive dynamic optimization routines in their current form take a prohibitive amount of time to calculate and ultimately result in erroneous flight path predictions due to inherent execution time latencies. Therefore an iterative, suboptimal heuristic approach was explored to mitigate excessive calculation times and ultimately yield improved flight path predictions. This report not only explores the heuristic techniques used for flight path calculation, but also includes real world application and flight test results in a Micro Air Vehicle equipped with an autopilot.

Acknowledgements

I would like to begin by thanking my thesis advisor, Dr. Cobb, for his perspicacity, patience and longsuffering with me on this project. I would also like to extend my deep appreciation for Major Blue's help along the way. His insights were indispensable, and his courses were some of the most rewarding taken here at AFIT. I would also be remiss to not thank my finance teacher, LtCol Smith, for enduring my rants on the folly of fiat currencies and need for sound money; hopefully you pondered gold investments as a result.

I would like to thank my wife for providing ballast in these often turbulent waters. You are truly God-sent. Thank you for loving a red-headed stepchild like me, when you could have done much better. Thank you for putting up with my late nights, and for your selfless devotion to me and more importantly our kids. You have opted to stay at home and teach the girls when you could have been painting nails or making money. In doing so you define leadership and agape love, and that is why I love you all the more.

I thank God for my precious girls. Lord, help me raise them to be joyful in all circumstances and women of Godly character. May I some day prove worthy of their undying adoration. I love you all.

The grass withers, the flower fades, but the word of our God stands forever.

Isaiah 40:8

The fear of the LORD is the beginning of wisdom: a good understanding have all they that do His commandment. His praise endures forever.

Psalms 111:10

To my helpmeet and closest friend: I love you.

To those who shed their blood to give us liberty: thank you.

"Guard with jealous attention the public liberty. Suspect everyone who approaches that jewel."

Patrick Henry

"It is a universal truth that the loss of liberty at home is to be charged to the provisions against danger, real or pretended, from abroad. "

James Madison

"If ye love wealth better than liberty, the tranquility of servitude more than the animating contest of freedom—go from us in peace. We ask not your counsels or your arms. Crouch down and lick the hands which feed you. May your chains sit lightly upon you, and may posterity forget that ye were our countrymen!"

Samuel Adams

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Symbols

$Line_{fwd}$	≡	Forward line created by the sensor Field-of-View (FOV)
$Line_{rear}$	≡	Rear line created by the sensor FOV
h_{uav}	≡	height (altitude) of UAV
$\theta_{\text{camElevation}}$	≡	Camera Elevation Angle
$\theta_{\text{camAzimuth}}$	≡	Camera azimuth Angle
θ_{camFOV}	≡	Camera FOV angle
R_{fwd}	≡	Slant range to forward FOV line
y_{sensor}	≡	vertical distance to optimal sensor location within FOV
V_w	≡	Wind Velocity
θ_{wind}	≡	Heading that wind is blowing from
W_x	≡	x component of wind vector
W_y	≡	y component of wind vector
V_t	≡	Target velocity
θ_t	≡	Target heading
V_{t_x}	≡	x component of target velocity
V_{t_y}	≡	y component of target velocity
x_{t_0}	≡	initial x-position of the target
y_{t_0}	≡	initial y-position of the target
x_{t_e}	≡	expected future location of the target in the x plane
y_{t_e}	≡	expected future location of the target in the y plane

θ_{uw}	≡	wind-corrected heading required to intercept target
V_{uav}	≡	UAV airspeed velocity
$V_{uav_{wx}}$	≡	x component of UAV groundspeed
$V_{uav_{wy}}$	≡	y component of UAV groundspeed
$V_{uav_{ground}}$	≡	UAV ground speed

List of Abbreviations

OEF	OPERATION ENDURING FREEDOM
OIF	OPERATION IRAQI FREEDOM
NLOS	Near Line of Sight
AFSOC	Air Force Special Operations Command
CP3	AFRL Core Process Three
JCIDS	Joint Capabilities Integration and Development Systems
BLOS	Beyond Line of Sight
HIL	Hardware in the Loop
TACMAV	TACtical MAV
GPS	Global Positioning System
INS	inertial navigation system
COTS	Commercial Off-the-Shelf
TOF	time-of-flight
ATR	Autonomous Target Recognition
GUI	Graphical User Interface
MAV	Micro Aerial Vehicle
OEM	Original Equipment Manufacturer
SDK	Software Development Kit
UAV	Unmanned Aerial Vehicle
VFOV	Vertical Field of View

REAL-TIME FLIGHT PATH OPTIMIZATION FOR TRACKING
STOP-AND-GO TARGETS WITH MICRO AIR VEHICLES

1. Introduction

1.1 Thesis Introduction

Lessons learned from OPERATION ENDURING FREEDOM (OEF) and OPERATION IRAQI FREEDOM (OIF) have demonstrated a need for teams on the ground in urban environments to organically engage high value, time-sensitive targets in real-time, from Near Line of Sight (NLOS) ranges (500m to 5km) without waiting for outside air support. Currently, engaging these NLOS targets requires coordination of orbiting assets such as fighter or bomber aircraft, Hellfire missile equipped Predator UAV's, or joint ground-based artillery systems. While devastatingly effective, these systems have three drawbacks: 1) they must be on-station and available for tasking at the time of the request, 2) they have a high probability of causing significant collateral damage, and 3) it takes time to pass the target information and receive clearance to engage the target – an unacceptable delay when engaging a fleeting, high value target.

These drawbacks have naturally led to research in the areas of man-portable Micro Aerial Vehicles (MAV's) which can be deployed by members on location, requiring minimal deployment time and minimal outside coordination. The thesis described herein is therefore the progeny of AFIT's response to the need of such a system. This thesis is the culmination of the theoretical work of several other groups, the implementation of some previous ideas as well as design and integration of new methodologies determined as best suited for completion of the task of creating a miniature, mobile system which can track and engage a moving target.

1.2 Problem Statement

Current man-portable weapon systems require the operator to have unobstructed Line of Sight (LOS) to the target at effective ranges less than 1000 meters. Maintaining an unobstructed LOS in an urban environment, while staying behind protective cover, is challenging at best. These current weapon systems also require the operator to partially expose themselves both giving their location away and exposing them to enemy fire. In addition, these weapons are essentially large, explosive bullets with no loiter or wait capability. With the current capability, a team tasked with engaging a time sensitive target in a small city would have to infiltrate the city undetected to within approximately 200m of the target. Assuming the ground team stayed covert while moving to intercept the target, detection is almost assured once the current weapon systems are used to engage the target. Given their distance from friendly forces, the survivability of the engaging team at this point would be very low. Further, if the target moves, or the ground force team is re-directed, their response time is comparably higher than other systems due to the team's need to stay covert while navigating through an urban environment.

The desired organic capability is a responsive, man-portable, self-propelled, low signature, expendable delivery system with loiter capability, and a NLOS range greater than that currently provided by fielded systems. This new system would allow the user to covertly launch, loiter, track, positively identify (ID), and engage a time-sensitive, high value target from a safe distance. The system should be effective in urban environments as well as desert, maritime, and temperate environments.

This thesis will concentrate on creating a flight path algorithm that can be implemented real time with minimum computation lag and will effectively maneuver a UAV sensor over a moving target. It will concentrate on developing the UAV flight path planning algorithms necessary to guide the UAV from launch to the target. It will specifically look at heuristic, iterative techniques which can quickly calculate flight path solutions, implement these solutions on actual UAV systems, and validate the algorithm through flight tests.

1.3 Scope and Assumptions

The following assumptions were made during the course of this research:

1. The aircraft is modeled as a point mass
2. The optimal sensor position is modeled as a point on the ground
3. The aircraft is flying at constant airspeed
4. The aircraft maintains a constant altitude
5. The wind direction and magnitude is constant for a given flight path solution
6. The target is assumed to be moving at a constant groundspeed for a given flight path solution
7. Sensor pointing with respect to the aircraft is fixed

This thesis focused on producing a robust flight path-making algorithm that can be implemented in a physical hardware system. The primary purpose of the developed routine is to plot an optimal path in the initial stages of the encounter, from time of launch until the target is detectable within the FOV. Once the target is within the FOV, tracking will be transferred to a separate Cursor-on-Target (CoT) tracking function (not

developed here). It uses hardware that is similar in size and capability to that used in other MAVs on the market. The initial testing occurred on a larger UAV test bed platform. After initial testing of the system is complete, the hardware was transitioned into a smaller MAV system. For the purpose of this research, a MAV is defined as having a wingspan of 24 inches or less, weighing less than 2 pounds.

1.4 Thesis Outline

In this chapter, an introduction to the concept of developing an optimal flight path model for a moving target was given, as well as the genesis of that requirement. Chapter 2 reviews the relevant work of previous AFIT students and others in this field of research, and their applicability to the current project. Chapter 3 details the implementation of theories discussed in Chapter 2, as well as the development of new systems and methodologies to accomplish the project goal. Chapter 4 details flight test results for the system, and provides analysis of these results. Chapter 5 concludes the thesis and gives pertinent recommendations for future work. An appendix is also provided that includes Matlab algorithms, flight test procedure outlines and hardware setup instructions.

2. Literature Review

2.1 Introduction

There is a growing volume of research dealing with tracking targets and optimal flight paths for UAS's. This Chapter provides a brief overview of AFIT theoretical research, hardware testing and procurement completed in advance of our work and will show in what ways this work is used in our current project. There were four individuals specifically who's work was directly related to the current thesis. This chapter will discuss the details of each of these theses that are relevant to the current project.

2.2 Sig Rascal as a Test bed Platform – Jodeh

Pivotal to any future UAV research at AFIT was the validation/verification of a viable flying test bed which could be used for UAV research. At the beginning of 2006, this capability did not yet exist. To this end, Nidal Jodeh conducted a thesis in March, 2006 to accomplish just this:

> The research in this thesis describes a UAV research platform developed to support the ANT Center's goals. This platform is now the bedrock for UAV simulation and local flight test at AFIT. The research has three major components. The first component includes development of a physical, inertial, and aerodynamic model representing an existing aircraft. A systematic analysis of the airframe leads to a complete geometric, inertial, and aerodynamic representation… Autonomous flights tune the autopilot controller through waypoint tracking in preparation for future advanced navigation research and provide data for Hardware in the Loop simulation validation. This report, along

with other significant legacy documentation and procedures, builds the foundation on which AFIT and ANT Center UAV simulations and flight tests are based. (Jodeh, 2006).

Capt Jodeh selected the Sig Rascal 110 as his primary test bed. This aircraft can be seen in Figure 1 below. A list of pertinent statistics for this aircraft are given below in Table 1. Capt Jodeh selected this particular airframe for the following reasons:

> It was chosen for its cavernous interior, stable flight characteristics, and use by other research institutions around the United States. The University of California-Berkeley's Center for Autonomous Intelligent Networks and Systems and Center for Collaborative Control of Unmanned Vehicles has utilized up to three of the Rascals in research of vision based road following using small autonomous aircraft (Ryan, et al, 2005:1). Dr. Eric Frew from the University of Colorado, and formerly from the University of California-Berkeley has also conducted vision based navigation work with the SIG Rascals (Frew, et al, 2004:1). The Rascal 110 is a high wing, tail wheel configured aircraft, commonly referred to as a tail dragger. The manufacturer constructed the aircraft of plywood, balsa wood, aluminum, and fiberglass and covered it with polyester film. Additionally, with a few modifications, the aircraft was upgraded with a 50 oz fuel tank, pushing the flight time to nearly two hours. (Jodeh, 2006:9)

Figure 1. Rascal 110's Out for Preflight Checks

Table 1. Rascal Physical Dimensions (Jodeh, 2006)

NAME	SYMBOL	VALUE
Wing Reference Area	S_W	10.56 ft^2
Wing Span	b_W	9.16 ft
Aspect Ratio	AR	7.94 ft
Wing Root Chord	c_{RW}	1.33 ft
Horizontal Tail Area	S_H	1.99 ft^2
Horiz. Tail Span	b_H	3.04 ft
Horiz. Tail Root Chord	c_{RH}	0.833 ft
Vertical Tail Area	S_V	0.773 ft^2
Vert. Tail Span	b_V	0.937 ft
Vert. Tail Root Chord	c_{RV}	1.0833 ft
Aircraft Mass (Empty)	M	0.4895 Slugs

The autopilot system that was used by Capt Jodeh was the Piccolo II airborne Avionics Package, seen in Figure 2. The Piccolo autopilot system has been used in other UAV platforms (Vaglienti, Hoag, and Niculescu, 2005:7), and was seen as an excellent test bed for future UAV research.

Figure 2. Piccolo II Airborne Avionics Package (Vaglienti et al., 2005:9)

As part of Jodeh's thesis, he also developed UAV physical, inertial, and aerodynamic simulation model of the Rascal 110. The wireframe representation of this model can be seen in Figure 3. Jodeh described his aircraft modeling:

> An accurate representation of the Rascal 110 in the Matlab/SIMULINK and Hardware in the Loop (HIL) simulations required precise physical and aerodynamic data. Physical measurements, CG, inertia, throttle and control surface calibrations were performed manually in the lab. While the airfoil model, engine and propeller model, and stability and control derivatives were performed using analytical software. (Jodeh, 2006:38)

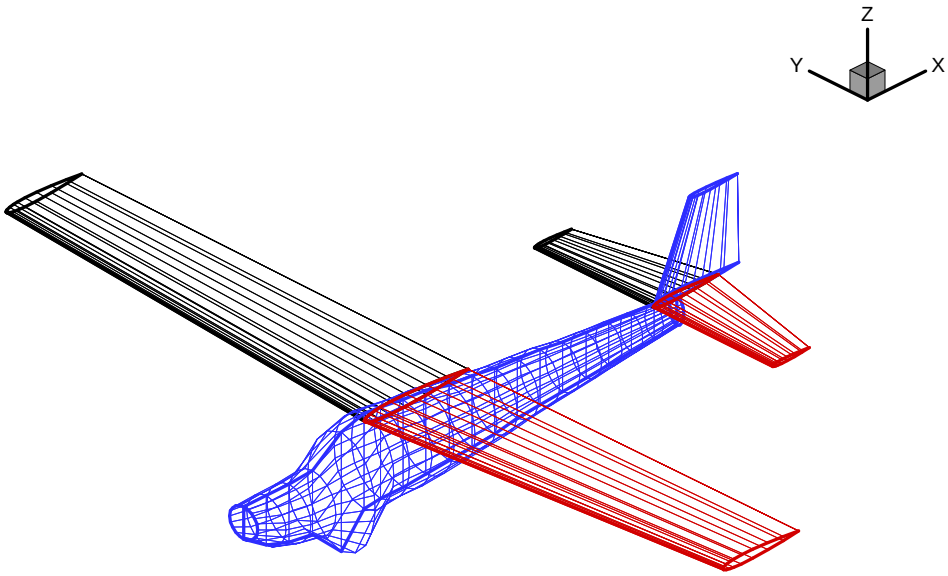


Figure 3. Rascal 110 Representation; Input to Digital Datcom (Jodeh, 2006)

The ultimate conclusion of Capt Jodeh's work was that he was able to develop a viable UAV platform for the AFIT ANT laboratory which has proved to be of great value for future research. By collecting physical, inertial and aerodynamic data for the Sig Rascal 110, he was able to get an initial estimate on correct autopilot values as well as develop a Hardware in the Loop (HIL) platform for the Piccolo autopilot.

The Sig Rascal was used as the primary air vehicle for all of the systems testing of the current thesis. It should be noted that the current thesis group voted against using the Piccolo autopilot used by Jodeh due to its size. The Kestrel autopilot system, which will be discussed later, was seen to better fit the vision for use with MAVs. Jodeh's work however was a crucial step in the procurement of the Sig Rascal as well as conducting the validation and verification of the airframe required for future autonomous research.

2.3 Wind Corrected Flight path planning – Zollars

Another major issue which must be addressed in any MAV flight planning is the effect of wind on the flight path. Not only does it affect the actual ground track, but the MAV heading can differ considerably from the ground track heading thus significantly affecting camera/sensor placement. Lt Michael D. Zollars (Zollars, 2007) conducted a thesis entitled "Optimal wind corrected flight path planning for autonomous Micro Air Vehicles", in which he studied minimizing the time-of-flight to position a MAV sensor over a specific point. Lt Zollars made the note that,

> Therefore, the goal of this research was to use dynamic optimization techniques to determine the optimal flight path to place a MAV's sensor footprint on a target when operating in wind for three different scenarios. The first scenario considered the minimum time path given an initial position and heading and a final position and heading. The second scenario minimized the error between the MAV's ground track and a straight line to the target in order to force a desired path on the vehicle. The final scenario utilized both a forward mounted sensor as well as a side mounted sensor to optimize the time the target is continually in view of the sensor footprint… These optimal flight paths provide a benchmark that will validate the quality of future closed-loop wind compensation control systems. (Zollars, 2007:iv)

At the root of Lt Zollars' thesis was the assumption that a TACtical MAV (TACMAV) would be used as the ultimate end-user platform. While the current thesis used the Sig Rascal as its primary test bed, the final airframe in mind was in fact identical to the one used in Lt. Zollars' research. The TACMAV also uses the Kestrel autopilot for

its autopilot system which became a determining factor in using Kestrel autopilot system for current research. Zollars describes the system:

> The Munitions Directorate of the Air Force Research Lab has developed a light-weight aircraft for military surveillance in the battlefield…This hand-launched vehicle is battery operated and navigates using a GPS/INS system. The vehicle utilizes two sensors for surveillance. One is mounted at a 45 degree angle directed out the nose and the other is mounted at a 30 degree angle directed out the wing.
>
> The lightweight TACMAV is equipped with sensors capable of viewing ground targets both directly in front, as well as out the side of the aircraft. However, this aircraft is very susceptible to wind disturbance which significantly decreases the ability to accurately point its sensor directly at the target at the desired approach angle. (Zollars, 2007:14)

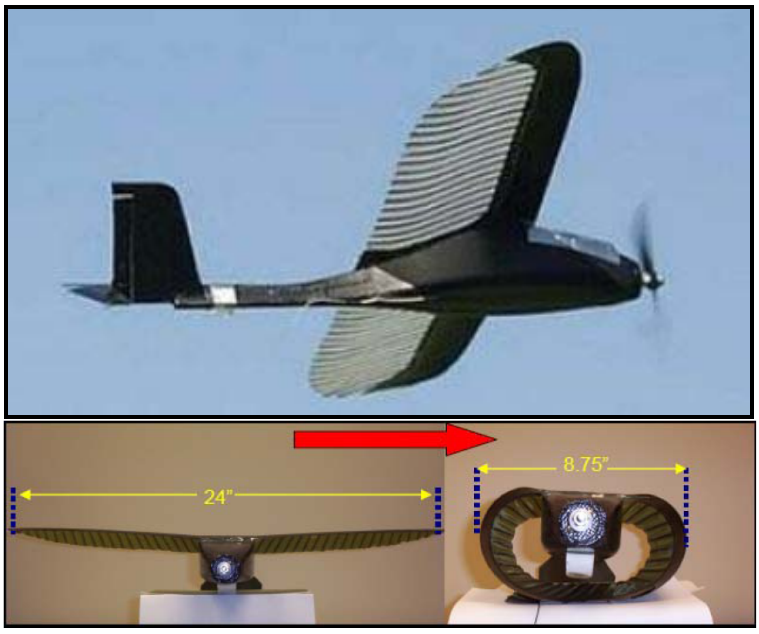


Figure 4. TACMAV Flexible Wings (Zollars, 2007)

Lt. Zollars focused on combing dynamic optimization techniques (Bryson, 1999) with Dubin's optimal flight path methods (Dubin, 1957). The dynamic optimization technique followed the standard form $\min_{u} J = \phi(x(N), t) = N t$. This standard dynamic optimization technique seeks to minimize the path cost, J, based on a series of weighted constraints along N number of steps. Zollars' work included path optimization techniques developed by Dr. Dubin:

> L. E. Dubin developed a method for determining the shortest distance between two points given an initial position and heading to a final position and heading…He showed that the shortest path taken will consist of not more than three pieces, each for which is either a straight line segment or an arc of a constant radius (Dubins, 1957). Many others have adopted his philosophy and used it in their own work. McGee, Spy, and Hedrick, out of the University of California Berkeley, determined optimal path planning in the presence of wind using Dubins methods (McGee, 2005). (Zollars, 2007:21)

Figure 5 depicts 4 potential solutions to a single optimal path. They consist of a executing a series of Right-straight-Right (RSR), Right-Straight-Left (RSL), Left-Straight-Left (LSL), or Left-Straight-Right (LSR) maneuvers. The turns are done at the maximum turn rate available to the aircraft. Using a parameter optimization algorithm such as Matlab's FMINCON function, these four scenarios were evaluated and the shortest time-to-target path selected. An example of Zollars' model results are given in Figure 6.

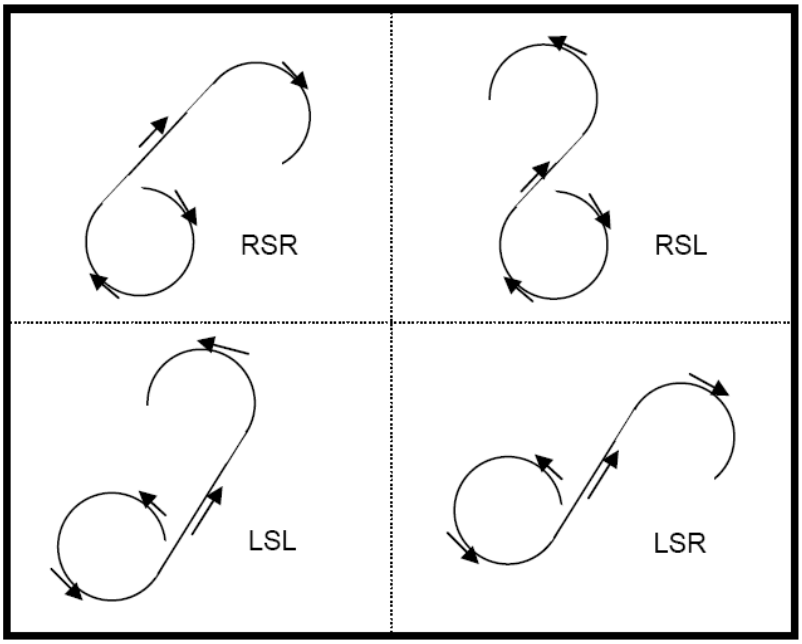


Figure 5. Four Examples of Admissible Dubin's Paths (Zollars, 2007:19)

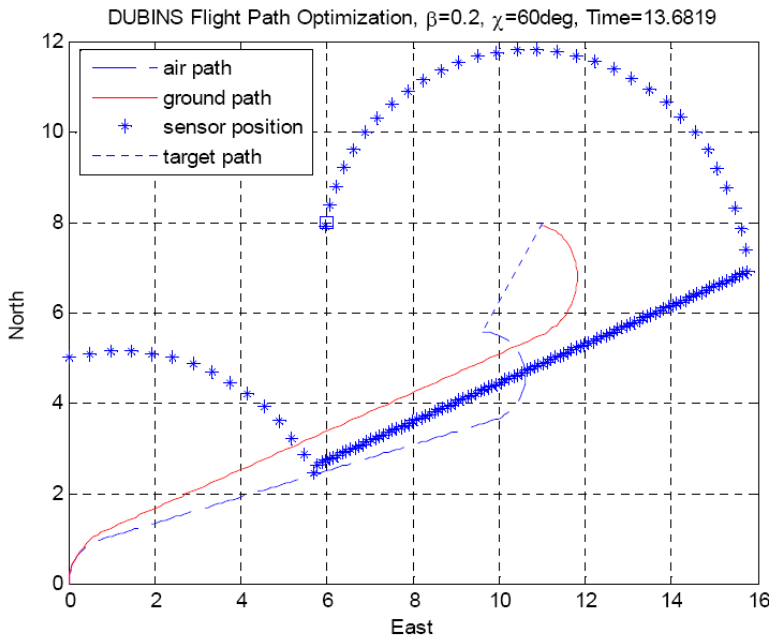


Figure 6. Dubins Flight Path with Forward Sensor (Zollars, 2007:31)

Lt. Zollars proved that optimization tools such as FMINCON could successfully be used to create an optimal intercept solution for a moving target in windy conditions. His work provided the starting point for the current thesis and an effective evaluation tool for future research. Ultimately Lt Zollars' FMINCON-laden approach to creating an optimal flight path proved too time-consuming and inefficient to be used in a real world scenario. It did, however, provide a conceptual basis for future work. His algorithms were also used to check the accuracy of future suboptimal solutions which were developed in this thesis.

2.4 Kestrel Autopilot Integration – Rufa

As has been previously mentioned, the group made a decision early on to incorporate the Procerus Kestrel Autopilot into the hardware configuration. Fortunately, some work had already been done at the ANT lab involving Kestrel autopilot systems by Lt Justin Rufa. The focus of Lt Rufa's thesis was target identification and recognition using MAV-sized hardware. Lt Rufa's research, "…demonstrates the ability of a given sensor to use a basic ATR algorithm to identify targets in a search area based on its size and color. With this ability, the system's target thresholds can also be altered to mimic real world UAV sensor performance." (Rufa, 2007:iv)

While the methods explored by Rufa for target recognition were not used in the current thesis, the hardware procured by the ANT lab for his thesis research served as the foundation for the current group's initial work. The RC dump truck, seen in Figure 7, served as the test bed and mounting platform for the autopilot and cameras. Its slow, variable operating speeds allowed for refinement of target recognition algorithms not

available in the Sig Rascal. It also added the additional safety and accessibility of a land-based system, where adjustments could be made easily. The dump truck was retrofitted with a plexiglas shelf; this allowed all the necessary hardware to be mounted onboard and readily accessible for programming and modification.

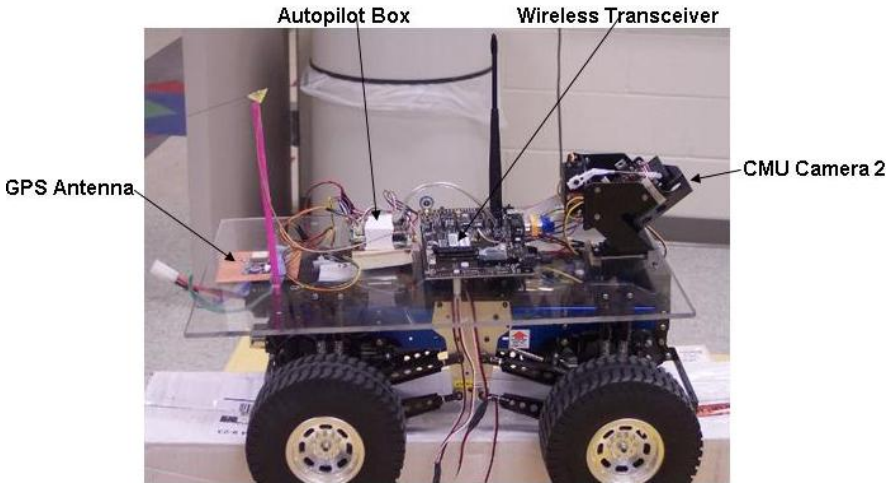


Figure 7. Dump truck with Plexiglas platform

The second major component used by Lt Rufa was the Kestrel autopilot system. He describes the system below (Rufa, 2007:24):

> The guidance for the search system comes from the Kestrel autopilot system, manufactured by Procerus Technologies in Vineyard, Utah (Procerus 2007). This autopilot provides the vehicle with its autonomous guidance and control ability with its GPS (Global Positioning System) and INS (inertial navigation system). The system is comprised of the actual onboard autopilot system and the ground station.

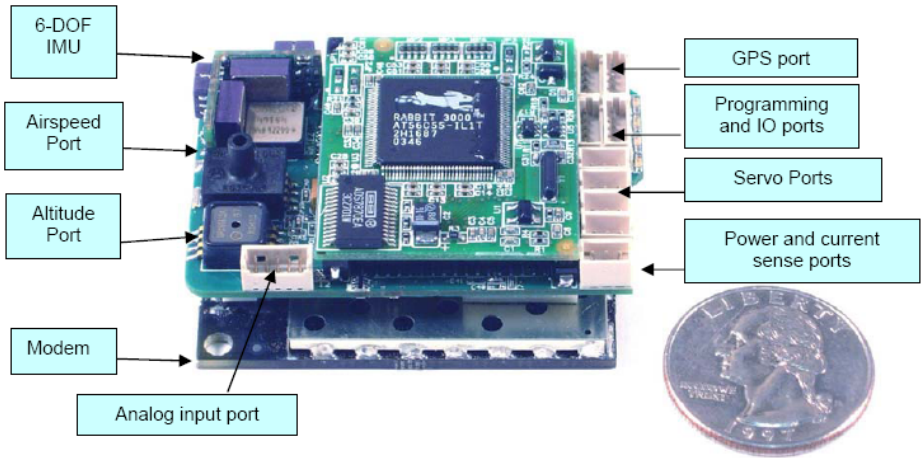


Figure 8. Kestrel Onboard Autopilot Box Input/Output Port Description,

The ground-based portion of the Kestrel Autopilot System consists of a Commbox receiver, RC transmitter, and the Virtual Cockpit software loaded onto a laptop computer. [as shown in Figure 9]. This ground station setup allows for all telemetry data to be relayed from the autopilot onboard the vehicle to the laptop via the Commbox through a RS232 9-pin serial cable. If manual control of the vehicle is needed, an RC transmitter can be connected to the Commbox and when configured properly the vehicle will respond to transmitter commands instead of autopilot commands from the ground station.

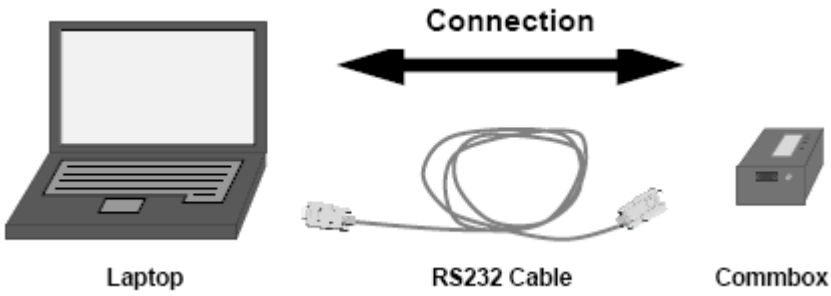


Figure 9. Kestrel Autopilot Ground Station Setup,

2.5 Real World Implementation of Optimal Path Planning

Other recent work in the area of optimal flight path planning has also noted the difficulty of implementing theoretical results into real world scenarios. A 2006 Conference paper from the Pennsylvania State University titled "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming" ran into this very problem (Greiger, 2006). Their optimal solution involved using nonlinear, receding horizon techniques and dynamic optimization cost functions that included maintaining the target in the field of view.

While their theoretical results proved promising, they admitted in their final conclusions that, "…the method is too slow for real time operation as it is currently implemented in MATLAB." Furthermore, they included in their conclusions that, "We performed two flight tests with an offline solution for a single UAV and stationary target, but the presence of a steady wind invalidated these offline generated flight paths." This implies that their optimal flight paths were calculated on the ground, but when applied to

the real world scenario the results were invalid. Creating feasible real world solutions is a primary goal of this thesis.

This chapter summarized some of the previous work that has been done to derive a viable real-time flight path optimization routine for use in UAV's. It also examined work previously done at AFIT that directly supported the current thesis work and paved the way for flight testing of optimal flight path algorithms on UAV platforms.

3. Methodology

3.1 Overview

This chapter lays out the fundamental processes for obtaining an optimal flight path solution for a moving target in a wind field. A simplified heuristic approach will be developed and compared to the optimal flight paths developed by Lt Zollars.

3.2 Wind-corrected Waypoint Generation

As discussed in Chapter 2, Lt Zollars previously did similar work to calculate the optimal flight path through a wind field to a stationary target. Zollars' work focused on using established dynamic optimization techniques in conjunction with Matlab's optimization toolset (Zollars, 2007). The result was an extremely accurate flight path algorithm that minimized time-of-flight to a stationary target.

When faced with implementing Zollars' theoretical results, a few items proved problematic. First, the computationally-intensive optimization code would often require in excess of 15 seconds to calculate the optimal path. Assuming the UAV is traveling a modest speed of 20 m/s, the UAV would now be 300 m from its original position. Because the algorithm assumed the UAV to still be at the point where the code was initiated, large positional errors were incurred as seen in Figure 10 . This figure illustrates that even though the original flight path prediction may have been extremely accurate, it is nonetheless useless as a final solution.

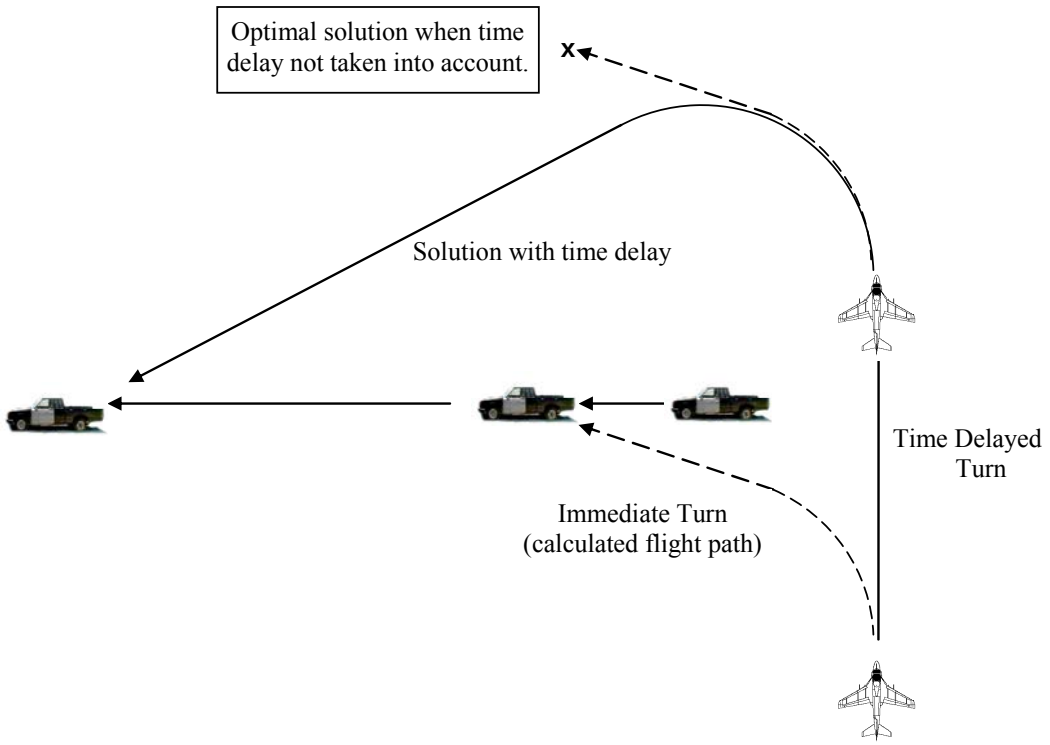


Figure 10. Effect of processing time on ultimate target location solution

Another problem was the fact that the amount of time required to calculate an optimal flight path was not linear or predictable. Zollars' traditional dynamic optimization technique relies on a dynamic optimization routine; this requires an initial guess for the flight path, and then subsequently tweaks the position of a predetermined number of stepping points to minimize the path cost. The dynamic optimization technique is highly dependent on the quality of initial guess, the size of the steps and the nature of the cost function. A greater number of step sizes, while generally taking longer to compute, does not ensure a more accurate result. Because the code execution time proved unpredictable, it proved impossible to extrapolate out the future position of the

aircraft to a point where the flight path commands would actually be executed. If, for example, we knew with relative certainty that it would take 10 seconds to compute an optimal flight path, we could effectively extrapolate the future location of the UAV, and optimize for that point. If, however, the calculation time is unpredictable and highly variant, no prediction can be made. The other option would be to force a return after a certain number of seconds. This would essentially guarantee an erroneous result of unknown tolerance if the optimization routine was exited prematurely, so this option was abandoned.

These two dilemmas persuaded the team to search for a better solution. One takeaway from Zollars' work is that, given a constant velocity wind field, the optimal flight path will consist of a maximum-effort turn to the target followed by a straight line course to the target (refer back to Figure 5). Utilizing these known assumptions a simplified Dubins-path method was designed to return a near-optimal solution, but with processing times of less than a second. As will be shown, the end result is a superior real-world solution through mitigation of the largest error-inducing aspect of the solution – that being the computation time required to determine the optimal path. The following sections will describe this process in detail.

3.2.1 Methodology

The suboptimal solution (in the future referred to as the Pathmaker solution) involves a heuristic approach. At its rudimentary level, the concept is rather straight forward. It is based on matching the TOF of the UAV to the intercept point (sensor on target), to the transit time of the target from its current position to the intercept. A set of iterations are implemented that predict the expected future location of the target and then

calculate the time required for the UAV to traverse the distance and place the sensor footprint on the target. The path course created by the UAV turning to the required heading to intercept the target is also calculated. By combining these two calculations, iterating through and getting increasingly accurate estimations on the Time of Flight (TOF) required to intercept the target, we can simultaneously estimate the expected location of the target vehicle with a high degree of accuracy. Figure 11 gives a top-level illustration of the process required to arrive at an optimized flight path.

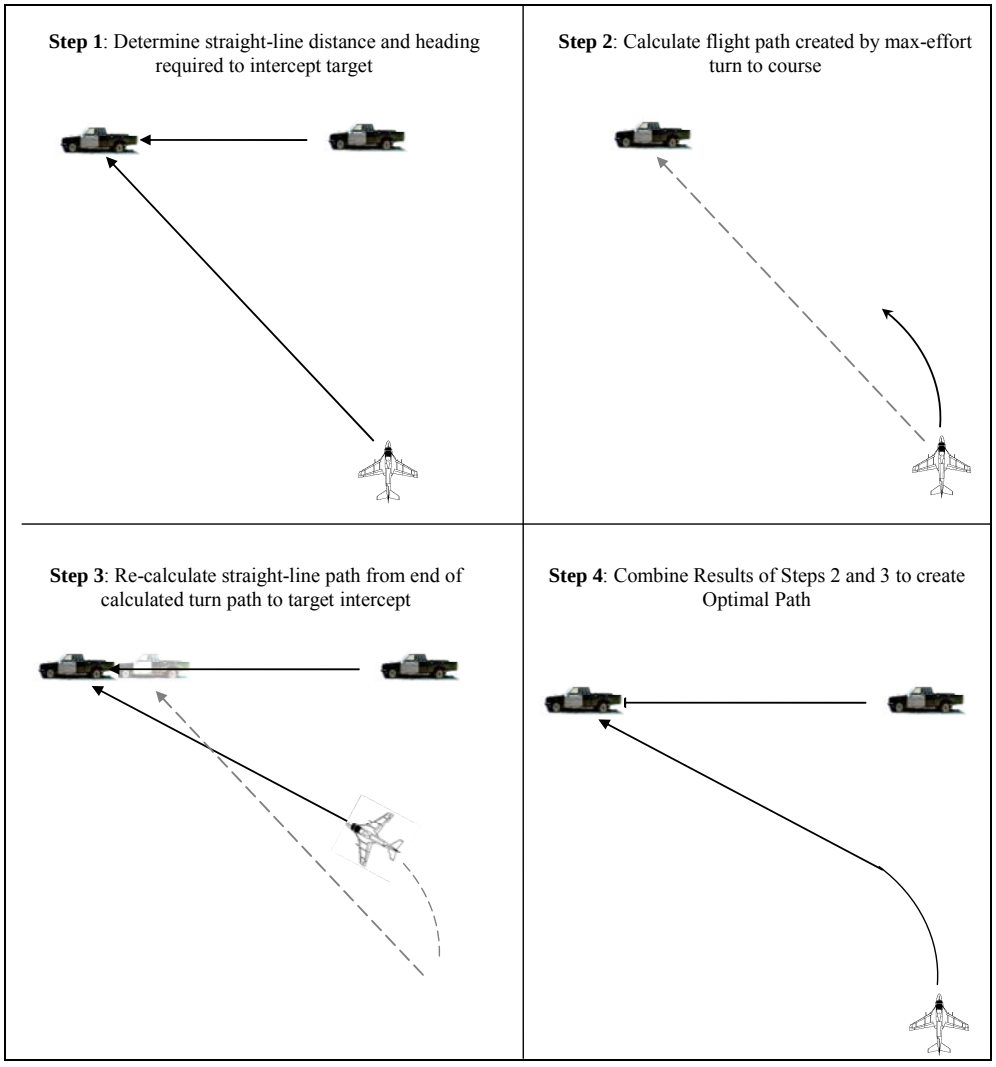


Figure 11. Steps required to produce flight path

The process for determining the future expected location of the target is done by iterating the calculated range to the target and the TOF necessary to intercept the target. A simple numerical demonstration of this process is given in Figure 12. For purposes of illustration, this figure assumes no wind and that the target vehicle is moving at half the speed of the UAV. With an initial target position 1 time unit away from the UAV, the target will move .5 time unit before intercept. As a result, the real TOF will be $\sqrt{1^2 + .5^2} = 1.118$ time units.

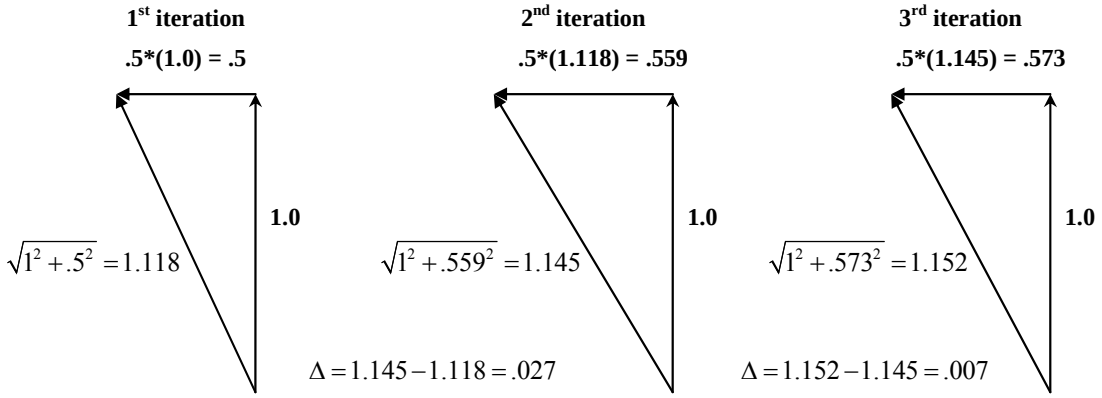


Figure 12. Numerical demonstration of straight line approximation

The key takeaway from Figure 12 is that the error (shown as Δ) in the expected future position of the target becomes smaller with each successive iteration. This is the underlying principle behind the iterative process and ultimately serves as an exit criteria out of the iterative loop. When the range approximations reach some predetermined tolerance of equivalency, the iterative loop is terminated.

Taking a further look at the process used to create the straight-line flight path approximation, a general step-by-step description of the process can be developed (refer to Figure 13). The numbers in parentheses are the equation numbers relating to the step procedure, which will be discussed later.

1. Calculate UAV sensor location relative to UAV (1-6)
2. Calculate the range between sensor and target (13)
3. Calculate TOF required to intercept target at current location (14)
4. Calculate expected location of target given current TOF (11,12)
5. Estimate heading required to intercept target, given wind speed and TOF (15)
6. Calculate necessary UAV final location, given wind correction and sensor footprint location (6)
7. Estimate ground speed given wind data(15-18)
8. Re-calculate range based new expected target position found in step 4 (11-14)
9. Iterate process until range estimations from steps (2) and (8) are equivalent (user-defined, but generally around 0.1 meter).

Each of these steps are described in more detail in what follows. Note that magnitude of change or error between range calculations in steps 2 and 8 is not used for any other purpose other than to determine whether the iterative loop should be continued; it is not used to construct a gradient or to determine which direction the optimization should continue. If the difference between the ranges is large, it means only that the process must be continued until the difference between range calculations is negligible.

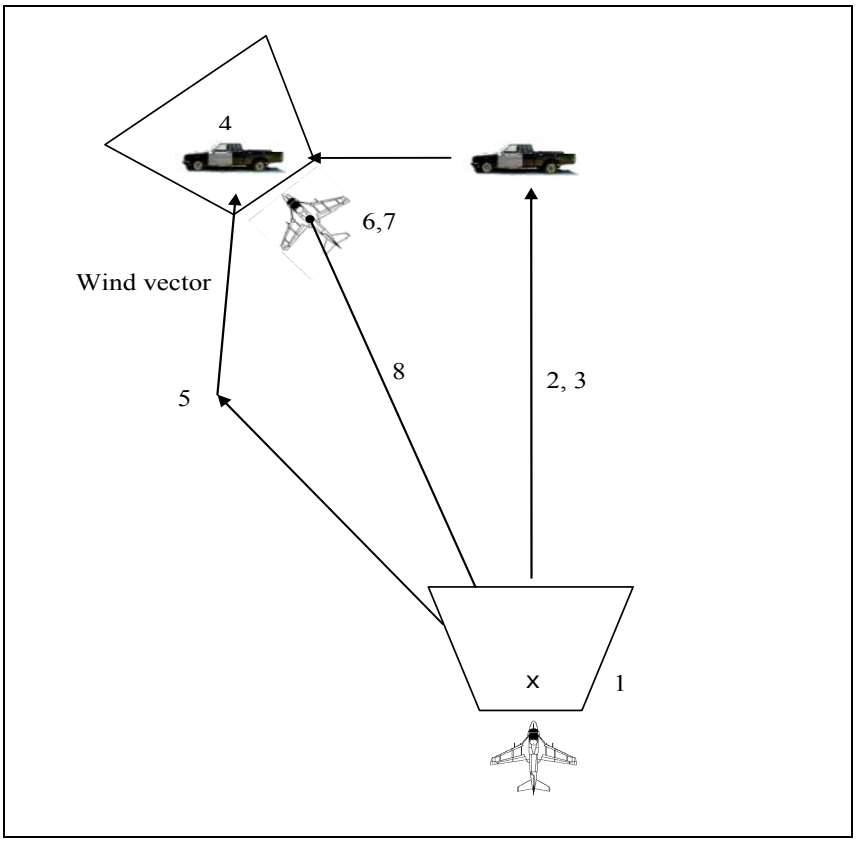


Figure 13. Illustration of iteration sequence of events

The process begins by assuming the UAV will intercept the target at its current position. This assumption is knowingly false, however it acts to seed the iterative process, giving an initial distance which can be used to estimate TOF and subsequent variables. In Figure 14 below, a generic example is given. This example would be indicative of a UAV flying in a northerly direction, a target moving in a southwesterly direction and winds from the south. As Figure 14 makes clear, the turn rate of the UAV is not taken into account at this time. Turn rate calculations are made in a higher level loop, and will be discussed in future sections. In the given example it can be seen that the initial estimate was quite erroneous, but successive iterations (blue lines) were relatively close to the final solution.

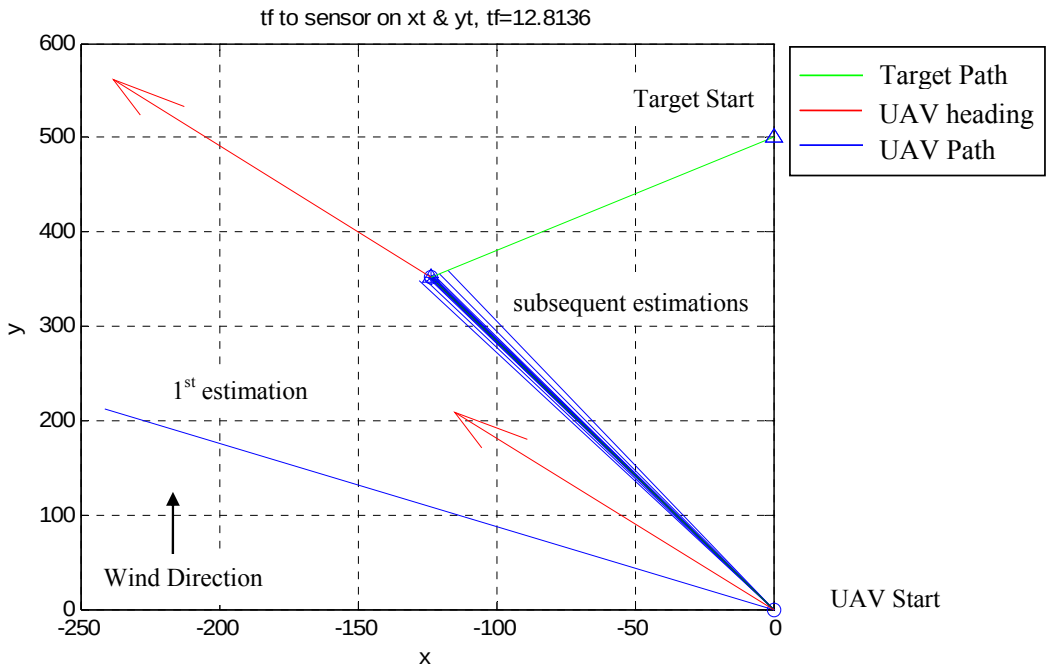


Figure 14. Example of standard iterative procedure

This straight-line approximation, which is the core of the optimization routine, is calculated both before and after any turn to course required for the UAV to intercept the target. It is calculated before the turn to course to determine what heading is required to for exiting out of the turn. It is calculated again after the turn to determine whether the expected location of the target has changed significantly since the initial calculation, as illustrated in Figure 11.

This lays out the basic methodology used to heuristically predict an optimal path. The next sections will go into further detail of specific aspects of the Pathmaker prediction.

3.2.2 Camera FOV and sensor footprint

To determine optimal positioning the UAV, it is necessary to know exactly where the sensor is looking on the ground. The camera field-of-view (FOV) is a projection of

the UAV's sensor trapezoidal footprint onto the earth's surface. This trapezoidal projection can be determined if the aircraft altitude and attitude are known, and given the camera elevation angle, azimuth angle, and camera FOV. These camera angles are illustrated in Figure 15.

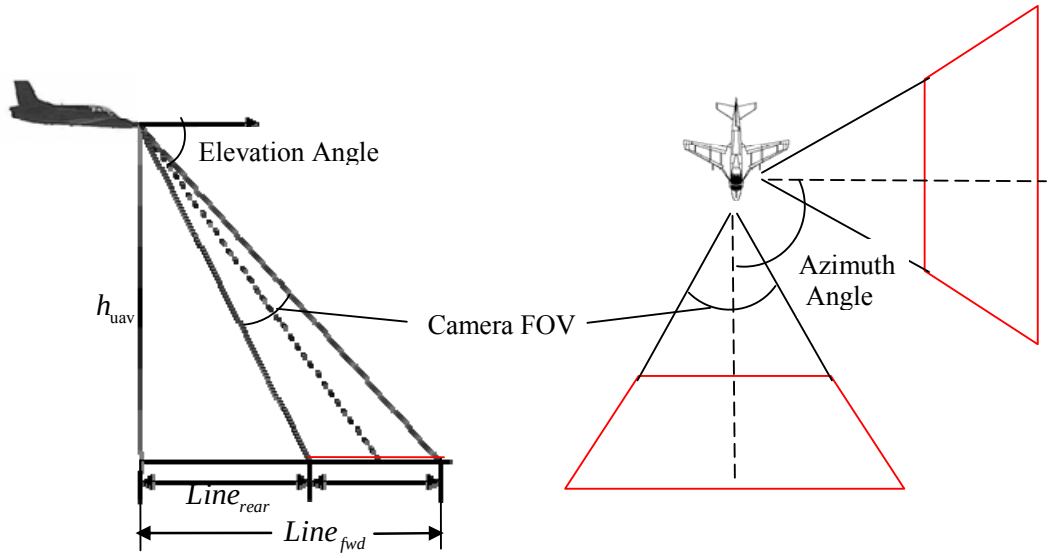


Figure 15. Camera FOV depiction

Referring to the vertical FOV above (LH illustration), the forward and rear limits of the FOV can be determined using the simple geometric equations below:

$$Line_{fwd} = \frac{h_{uav}}{\tan(\theta_{camElevation} - \frac{\theta_{camFOV}}{2})} \quad (1)$$

$$Line_{rear} = \frac{h_{uav}}{\tan(\theta_{camElevation} + \frac{\theta_{camFOV}}{2})} \quad (2)$$

To obtain the corners of the trapezoid, one must take into account the normalized distance from the aircraft to the corner points. With the front and rear FOV limits known, we can calculate the corners using Equations 3-5 (see also Figure 16). Note that the rear FOV limits are found in a similar fashion as those described for the forward FOV below.

$$R_{fwd} = \sqrt{h_{\text{uav}}^{\ 2} + Line_{fwd}^{\ 2}} \tag{3}$$

$$Cam_{fwd\_LHS} = R_{fwd} \cdot \tan(\tfrac{\theta_{\text{camFOV}}}{2}) \tag{4}$$

$$Cam_{fwd\_RHS} = -R_{fwd} \cdot \tan(\tfrac{\theta_{\text{camFOV}}}{2}) \tag{5}$$

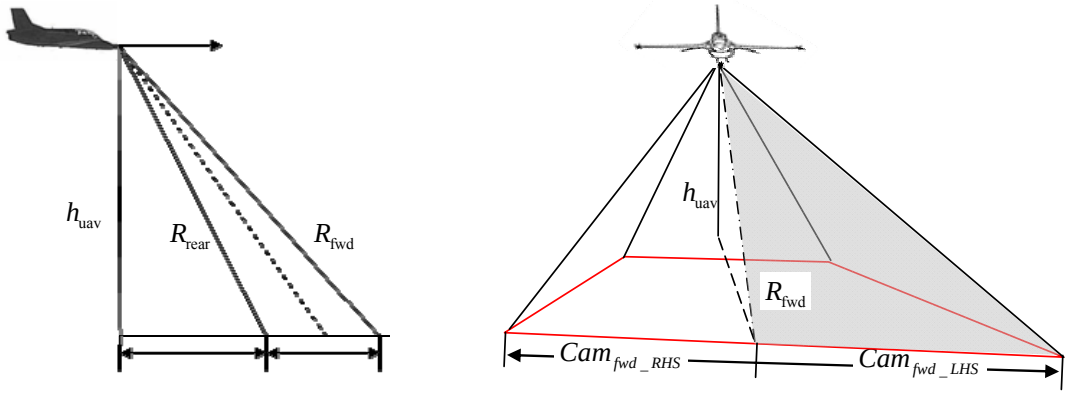


Figure 16. Determining corner points for camera FOV

Pathmaker uses a single point for sensor-on-target determinations. Pathmaker uses the sensor bore site as the optimal sensor positioning location. With this being the case, then the sensor point used in steps 1,6 and 8 in the iteration process will simply be:

$$y_{sensor} = \frac{h_{uav}}{\tan(\theta_{\text{camElevation}})} \tag{6}$$

One anomaly that arises in low slant angle (small elevation angle) is that upper FOV limit for the sensor may actually be above the horizon. Within the trapezoidal construct, the forward bound would be an infinite distance away. A white paper conducted at AFIT (Knowlan, 2005:27) provided some research on target detection and identification using UAV sensor platforms. This paper showed that using a standard 768x494 resolution camera (as was used in our studies), pixel density prohibits target detection of a standard vehicle sized target at ranges beyond 700 m. As a result, a limiting function was built into Pathmaker to clip the forward sensor FOV line. This line is adjustable, but a setting between 600-800m was generally used as the maximum viewable distance. Clipping this forward FOV line currently does not change the bore sight calculation used to determine optimal sensor position.

3.2.3 Wind correction

In order to make accurate calculations, a wind vector must be supplied. Wind data is derived by the Kestrel autopilot. Given a known airspeed (from pitot-static data) and GPS ground speed, the wind speed and direction can be readily derived. Wind data is part of the data packet received back from the autopilot and fed to the Pathmaker algorithm. Pathmaker assumes wind speed and direction to be constant between the target and the UAV. The wind vector is further broken down into X and Y components as follows:

$$W_x = V_w \cdot \cos(\theta_{wind}) \tag{7}$$

$$W_y = V_w \cdot \sin(\theta_{wind}) \tag{8}$$

3.2.4 Target Location Prediction

Pathmaker assumes that the target will be traveling at a constant speed and direction for the duration of the intended flight path calculation. From a systems integration perspective, a change in target course and speed will first be noted by the user. When the target course and speed are modified, it will automatically force a flight path re-calculation.

It should be noted that within the Pathmaker algorithm, all positions are relative to the UAV. At the system level, it is assumed that a valid lat/long are given to the user, who inputs that data into the system. This lat/long data is then converted at the system level into a relative position to the UAV. This data is conditioned into relative position prior to, and after being received back from, Pathmaker. Within Pathmaker, all calculations are made in metric units.

The target velocity used is broken into X and Y components using equations similar to the wind correction equations:

$$V_{t_x} = V_t \cdot \cos(\theta_t) \tag{9}$$

$$V_{t_y} = V_t \cdot \sin(\theta_t) \tag{10}$$

The future location of the target (x_{t_e}, y_{t_e}) is determined by estimating the TOF required to place the UAV's sensor on the target, multiplying this by the velocity of the target, and adding the vector to the current known position:

$$x_{t_e} = x_{t_0} + V_{t_x} \cdot TOF \tag{11}$$

$$y_{t_e} = y_{t_0} + V_{t_y} \cdot TOF \tag{12}$$

$$Range = \sqrt{(x_{t_e} - x_{sensor})^2 + (y_{t_e} - y_{sensor})^2} \tag{13}$$

$$TOF = \frac{Range}{V_{uav_{ground}}} \tag{14}$$

Notice that there is a circular reference between equations required to calculate the Range and TOF. This is why an iterative process is required. Through successive iterations on the range to the expected location of the target and UAV time of flight, the future vehicle location can be predicted with a high degree of accuracy.

3.2.5 UAV flight path prediction

The UAV location is determined using an iterative process similar to that described for the moving target. Given the target's course and speed, the expected location of the target at interception is calculated. Once this course is determined, a wind-corrected heading is calculated to place the sensor on target. The ground speed is also calculated and used to determine the TOF, using the set of equations below:

$$\theta_{uw} = \text{atan}\left(\frac{x_{t_e} - x_{sensor} + Wx \cdot TOF}{y_{t_e} - y_{sensor} + Wx \cdot TOF}\right) \tag{15}$$

$$V_{uav_{wx}} = V_{uav} \cdot \cos(\theta_{uw}) + W_x \tag{16}$$

$$V_{uav_{wy}} = V_{uav} \cdot \sin(\theta_{uw}) + W_y \tag{17}$$

$$V_{uav_{ground}} = \sqrt{V_{uav_{wx}}^2 + V_{uav_{wy}}^2} \tag{18}$$

Given the known maximum turn rate for the vehicle, Pathmaker calculates both right hand and left hand paths to the predicted necessary intercept heading in a technique similar to that used by Zollars (Zollars 2007:27). It then rejects the longer of the two

paths. An example of this procedure is demonstrated in Figure 17 with a UAV flying north, the target moving south, and easterly wind. The turn to course is an iterative process in itself which incorporates the course optimization routine described earlier. The iterative steps are as follows:

1. Estimate future position of the target given current range and TOF
2. Calculate the arcs created by a max-rate turn to both the RH and LH sides. The length of the arc is determined by the final heading required to place the UAV sensor on target.
3. Calculate elapsed time required to make the turn and fly to target.
4. Re-calculate estimated future position of target based on new TOF
5. Repeat

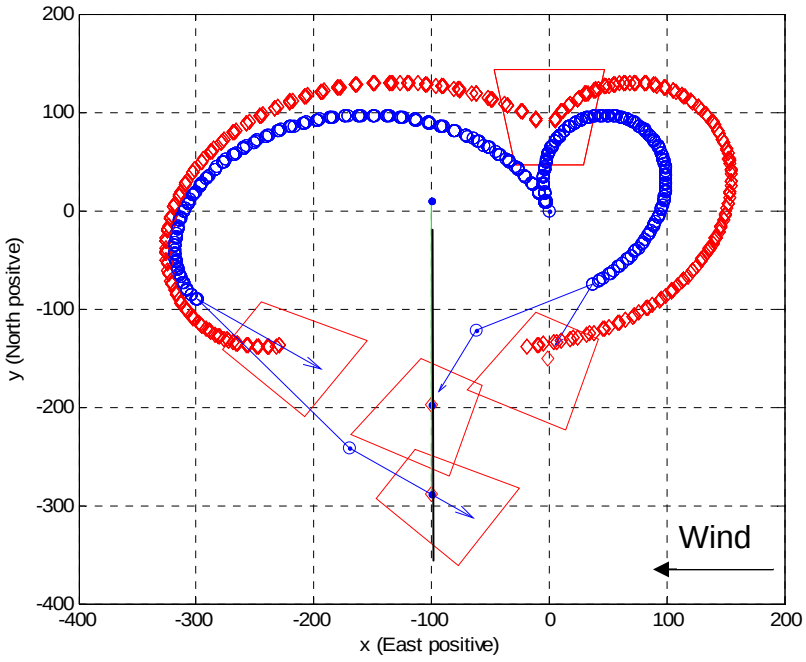


Figure 17. Determining shortest turn path to target

A latency term was built into Pathmaker to account for the time required to calculate an optimal path and send it to the UAV for execution. This latency factor estimates the future position of the UAV at the time of actual command execution by assuming the UAV will travel straight and level along the current vector for the duration of the latency time.

The time required for Pathmaker to compute a flight path is under a second, however due to modem transmission delays, the average time required for the UAV to begin execution of a new flight path is approximately 2 seconds. Figure 18 shows the effects of a 2 second and 30 second latency on the ultimate flight path of a UAV flying due north and intercepting a target heading west. The first obvious insight is that not taking into account a 30 second latency will provide a result that is wholly unusable.

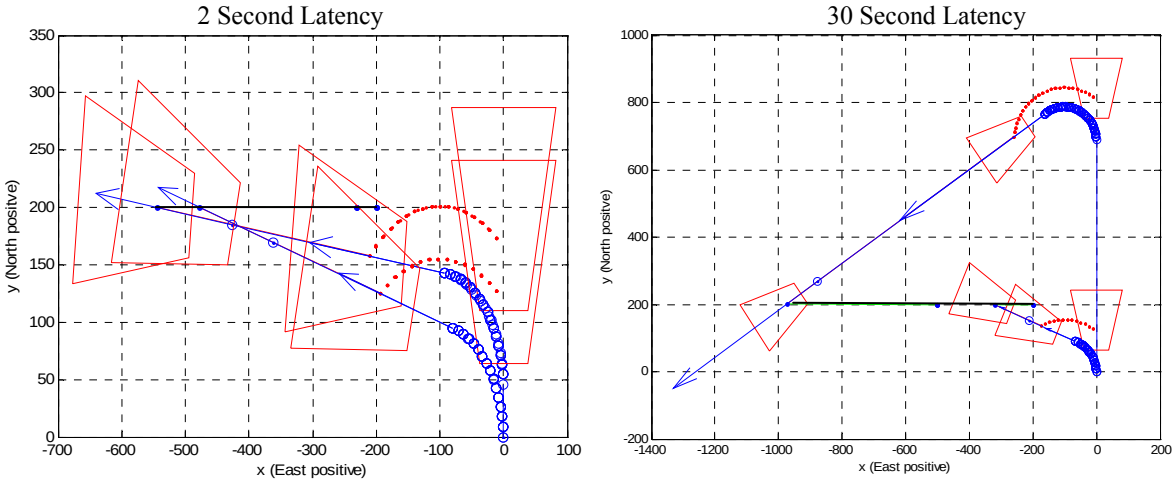


Figure 18. Effects of command execution delays on optimal path

Utilizing the above described techniques provides for a realistic, near-optimal flight path solution that closely approximates dynamic optimization techniques, as will be discussed next in the analysis results.

3.2.6 Analysis Results

NOTE: This section contains many graphs depicting calculations made using flight path optimization algorithms. To reduce clutter within the charts and the text, much of the initial conditions information was omitted. Please refer to Appendix A for a more complete set of graphs and tables containing all the information pertinent to each test case.

Within the proceeding charts, the symbology can be broken down as follows: the blue circles represent UAV waypoints or turn points. Blue or red arrows extending from the circles represent the UAV heading. A blue line represents the travelled path of the UAV. Red trapezoids represent the sensor FOV, and a red diamond represent the optimal sensor point. Target motion is depicted by the solid black line.

To determine the effectiveness of the Pathmaker routine, Pathmaker results were compared to Zollars' (Zollars, 2007) dynamic optimization routine. One of the problems with Zollars' method was that it would not always converge on an optimized solution. Selected cases were created to ensure that Zollars' method did in fact converge on an optimal solution. Figure 19 below demonstrates the high degree of correlation between Zollars' dynamic optimization and the developed Pathmaker routine. A stationary target is presented in the left hand side of Figure 19; the right hand side is a moving-target comparison. Table 2 includes the initial conditions for each of these runs as well as the final solution difference between the two methods. In the moving target case, the Pathmaker approach was within 8.5 meters of the optimal path, which is well within the standard sensor field of view.

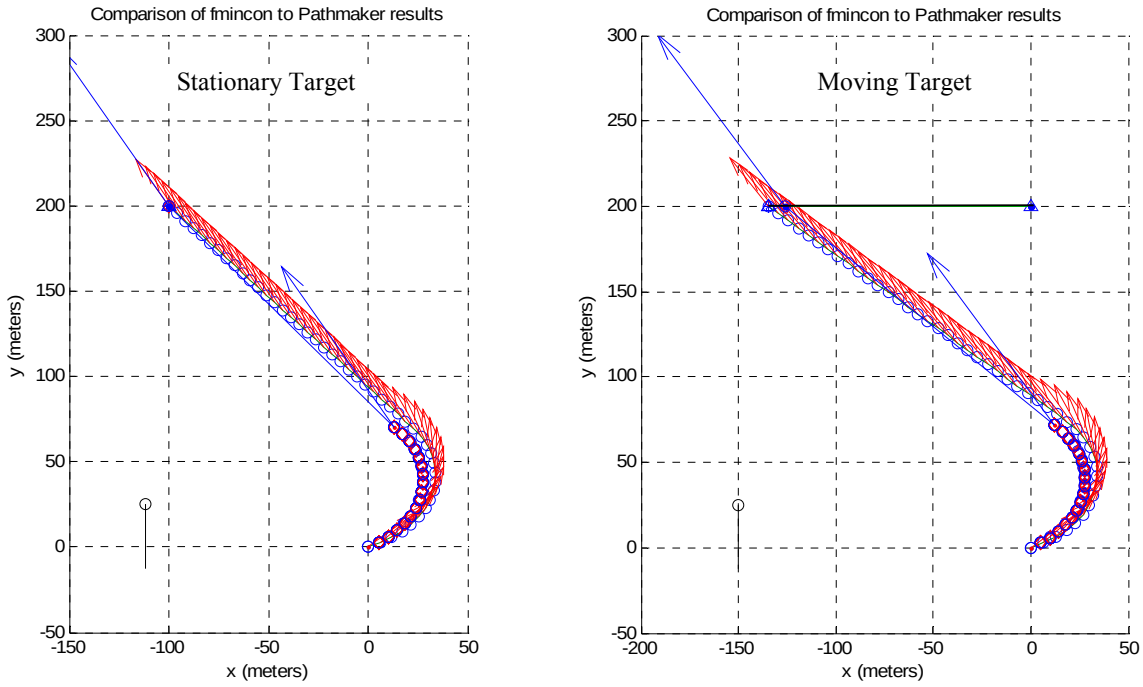


Figure 19. Comparative results with stationary and moving targets

Table 2. Run data for path comparisons shown in Figure 19

	Initial Conditions			Final Position		
	Target	UAV	Wind	Fmincon	Pathmaker	Difference
Initial X (m)	-100	0		-100	-100	0
Initial Y (m)	200	0		200	200	0
Speed (m/s)	0	30	10			
Heading (deg)	n/a	50	0			
Initial X (m)	0	0		-134.5	-126.04	8.46
Initial Y (m)	200	0		200	200	0
Speed (m/s)	10	30	10			
Heading (deg)	270	50	0			

Another figure of interest is shown in Figure 20. This figure shows two separate optimal solutions calculated with Zollars' algorithm, using two different step sizes. Note that the optimal path found with a step size (N) of N=60 is closer to the value found using the Pathmaker routine. This would indicate that the Zollars' optimization routine is actually converging on a trajectory similar to that calculated with Pathmaker. Upon

looking at Zollars' algorithm, one reason for the disparity is due to the nature of how the flight path is calculated between points in a turn. The Zollars routine assumes that the UAV will fly in a straight line between points. Therefore, in a solution with a course step size, the calculated turn radius will actually be larger than actual. We can see that this is exactly what is happening in Figure 20 below.

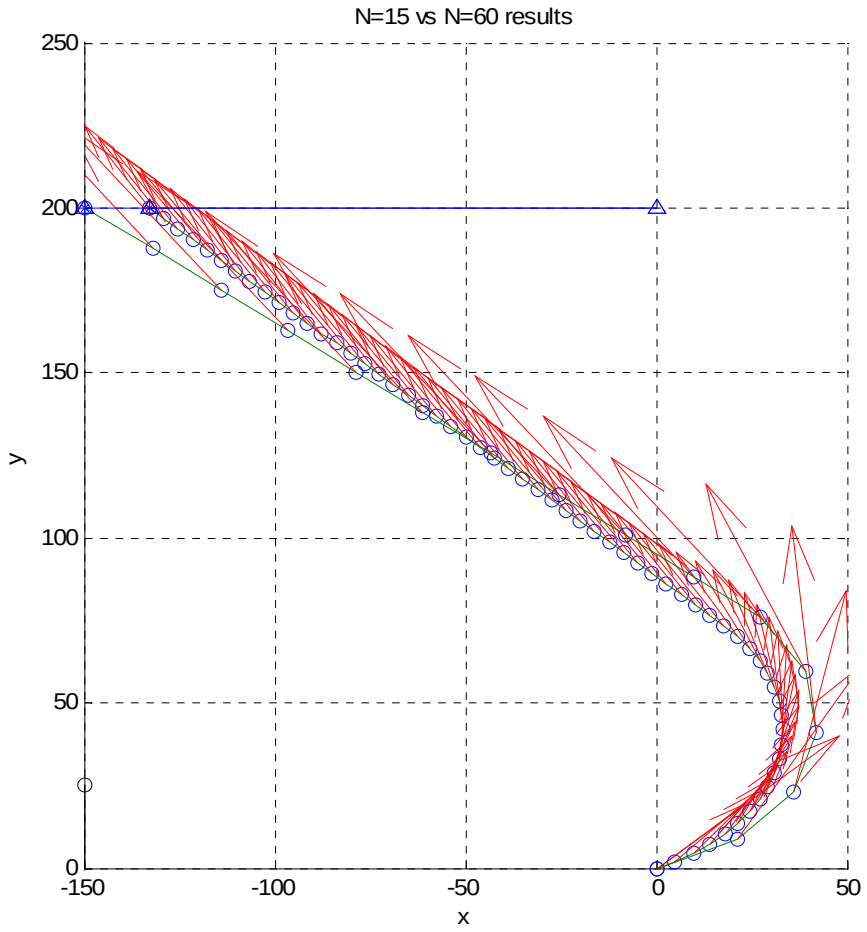


Figure 20. Effects of coarse step size on calculated turn radius

Another significant measure of the accuracy of the final solution can be demonstrated by including a representative computational latency into the flight path calculation. While Zollars' algorithm does not allow for this, we can examine a

representative latency using the Pathmaker algorithm. Figure 21 below gives an example of Zollars' optimized solution with a somewhat coarse step size of N=20. This specific solution required 8 seconds to calculate. The Pathmaker solution included in this figure demonstrates the effects of an 8 second latency on the UAV flight path and ultimate intercept location of the target. It should be noted that the calculation time required for the Pathmaker solution was 0.21 seconds. From Table 3 below we can see that the difference between the original optimized routine and Pathmaker-predicted target location is approximately 174 meters. This significant disparity illustrates that the Pathmaker flight path calculation offers a superior real-world estimation of the target location.

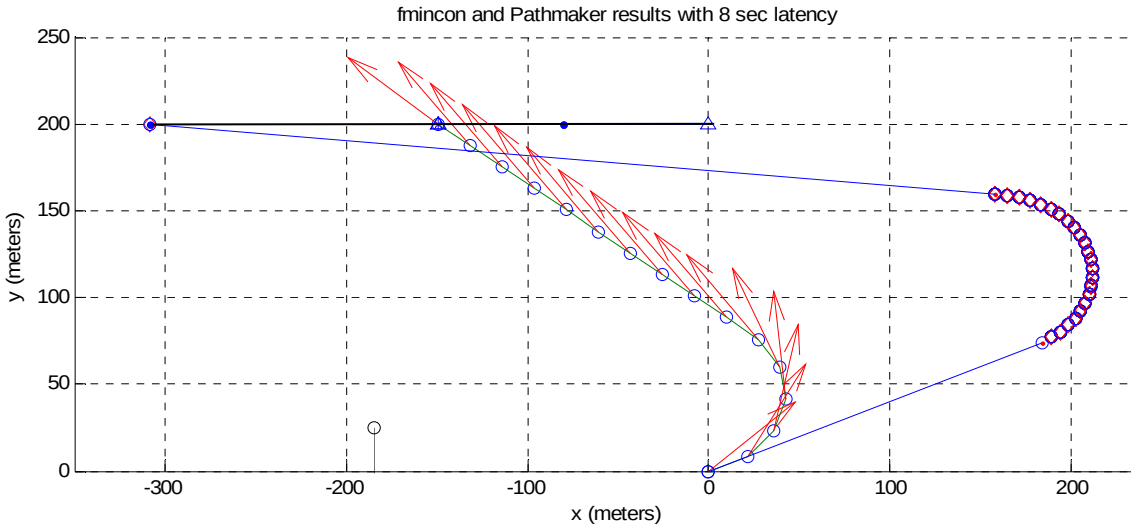


Figure 21. Effects of 8 second computation latency on final result

Table 3. Run data for path comparisons shown in Figure 21

	Initial Conditions			Final Position (m)		
	Target	UAV	Wind	Fmincon	Pathmaker	Difference
Initial X (m)	0	0		-134.5	-308.35	-173.85
Initial Y (m)	200	0		200	200	0
Speed (m/s)	10	30	10			
Heading (deg)	270	50	0			

The final and primary point of interest for using the Pathmaker algorithm is for variable winds and stop-and-go targets. Results could not be compared with Zollars' routine, as his algorithm was not built to accommodate such calculations. We will start by discussing stop-and-go targets. As was discussed earlier, an assumption is made within this thesis that target data is current and accurate, and that exact times for change of course and speed are known. This would be representative of the case where the target is initially spotted by someone either on the ground or in the air who can monitor when the target changes course. Given this assumption, the Pathmaker algorithm can be used to make real-time flight path modifications. The routine would proceed as follows:

1. Spotter calls in new course and speed for target to UAV operator.
2. UAV operator inputs target's new course, speed, and time of change into laptop.
3. Operator executes Pathmaker algorithm, and new flight path is calculated.

Figure 22 demonstrates the stop-and-go technique under three different camera scenarios. Each graphed solution used identical input variables except for the camera FOV used to calculate the solution. The top graph uses a forward looking camera scenario, where as the middle graph uses a side-looking camera configuration. The bottom graph uses a front-looking camera for the first two legs, and then switches to a side-looking camera for the final intercept. This is possible because each leg is optimized for the current camera configuration. The solitary blue circle with magenta asterix marks the position of the UAV at the conclusion of each 10 second interval, and beginning of each subsequent calculation.

The tests consisted of three legs. The first two legs were each 10 seconds long; the third leg was kept indefinite and determined by sensor placement on the target. With the target traveling north at 10 m/s, a stop was made at (-400, 100) for 10 seconds before continuation north again. A time lag of 2 seconds was used to demonstrate computational latencies. UAV airspeed was 20 m/s; wind was constant from heading 315 degrees at 5 m/s. As can be seen by these results, Pathmaker was able to successfully adjust to the stop-and-go target.

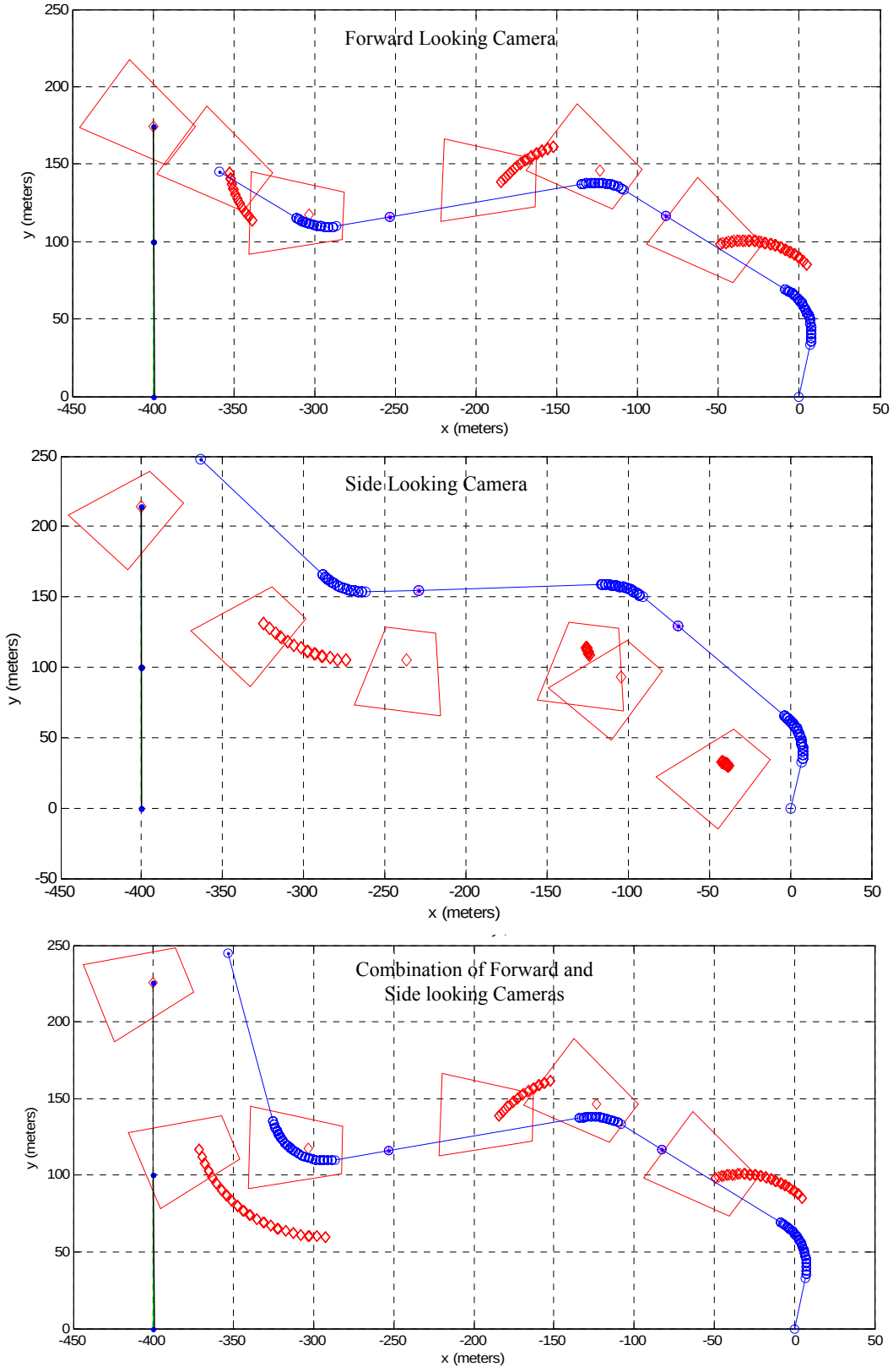


Figure 22. Tracking a stop-and-go target

A more difficult test of the system would be a target whose course and speed are changing. This would be analogous to a target that is actively trying to evade surveillance, or that may be traveling down a road. Figure 23 illustrates such a scenario, where the target course and speed are varied. This example also incorporates a 2 second time lag. The target originally travels north at 10 m/s, then east at 5 m/s, south at 20 m/s, and finally west at 15 m/s. UAV airspeed was 20 m/s; wind was constant from heading 315 degrees at 5 m/s.

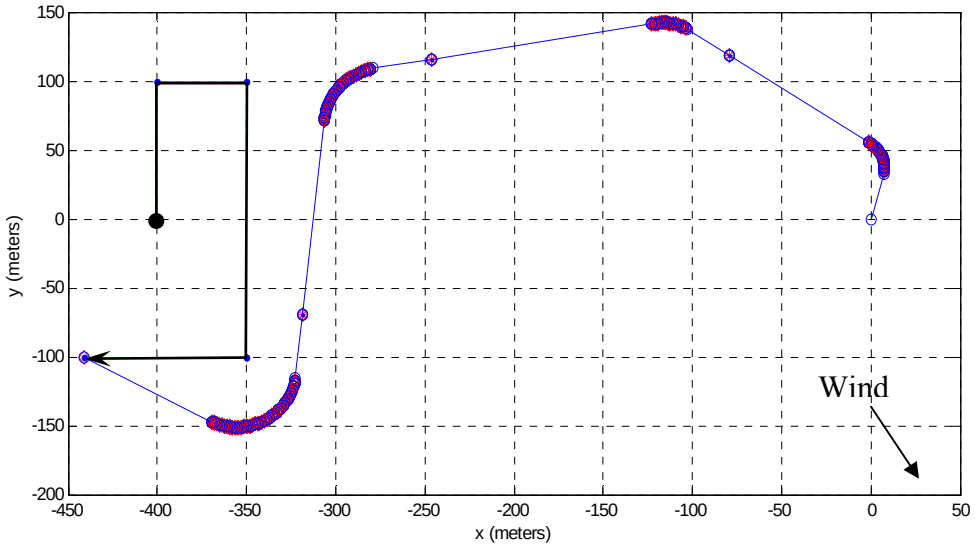


Figure 23. Tracking a target that has variable course and speed

The Pathmaker algorithm can also be used to make in-flight adjustments to variations in wind. In theory, this wind-correction process could be done automatically without any input from the user. Because the wind data is derived from GPS and pitot-static data, it is calculated onboard the autopilot and is part of the standard packet of information available. It is therefore possible to take the wind data as it is updated in real-time from the autopilot, and execute Pathmaker at any point the wind speed and

direction fall outside of a set tolerance. Furthermore, if a processor capable of running the Pathmaker solution was located onboard the UAV, this calculation could be done in real-time with time lags of less than a second. Currently all Pathmaker processing is done at the ground station where the wind data is transmitted to the laptop, Pathmaker calculates a new solution, and the new solution is uploaded back to the UAV.

Figure 24 illustrates a potential wind correction scenario that involves three legs. The wind speed remained constant at 10 m/s for each leg, however the direction changes as noted in the graph. A time lag of .5 second was used to demonstrate a scenario utilizing an onboard processor. Target velocity was constant at 10 m/s, north. UAV airspeed was 25 m/s.

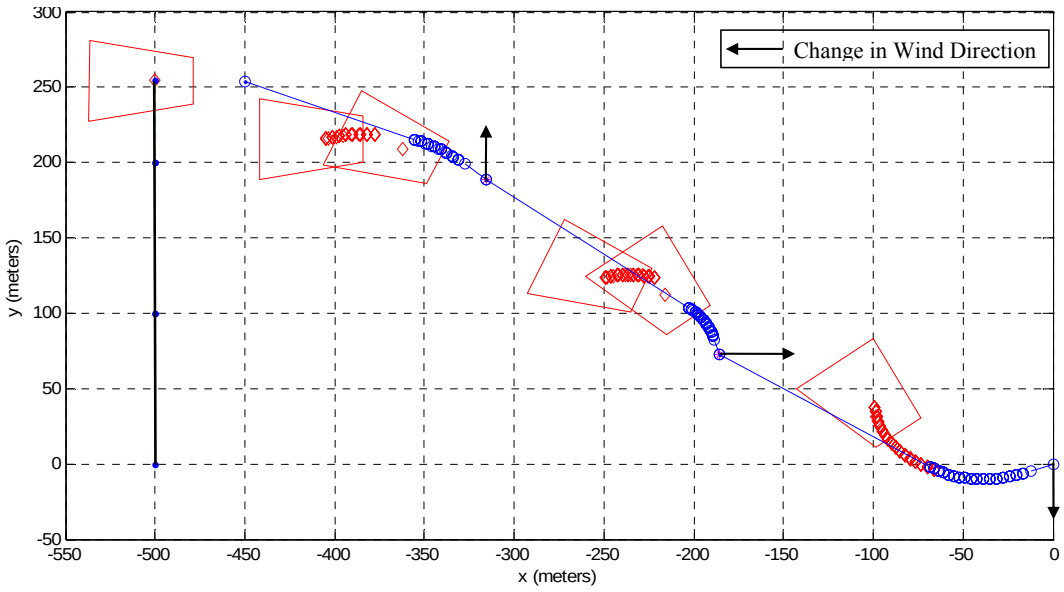


Figure 24. Tracking target with variable wind direction

3.2.7 Pathmaker Weaknesses

There are some inherent weaknesses, which, while not being exclusive to the Pathmaker algorithm, should be discussed. The ability to precisely track a target is

dependent on receiving timely information on the target. In a real world scenario, a target change of course will be relayed to ground station unit, entered into the laptop, and Pathmaker will be executed. This whole process when moving efficiently takes approximately 20 seconds. Should the target make any changes in course within this time window, it is easy to see that the operator will constantly be lagging behind the movements of target. An example of this is given in Figure 25. The closer the UAV is to the target, the more pronounced the effect of these time lags will be on accurately target tracking.

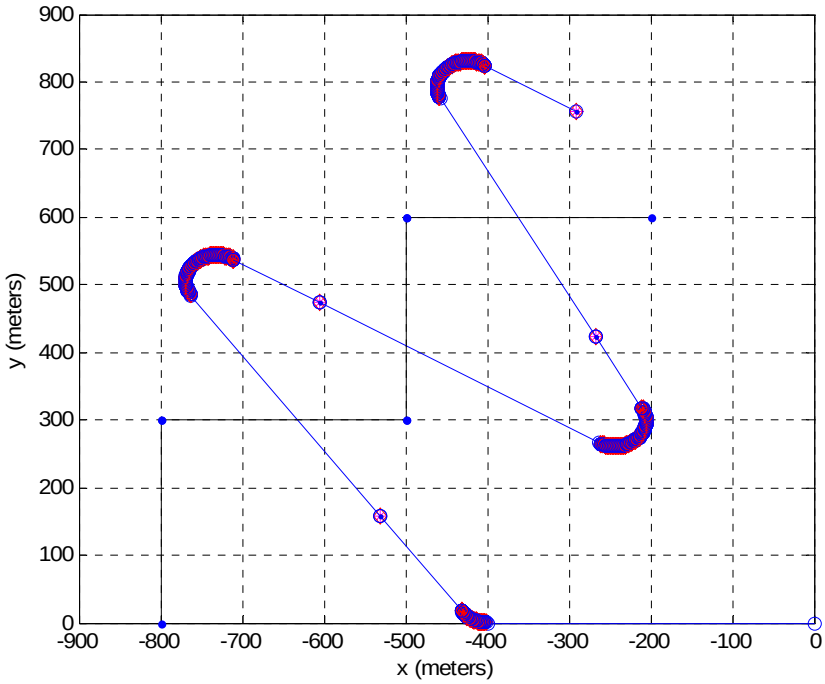


Figure 25. Tracking difficulties as result of high lag times

This weakness was recognized from the beginning. The primary purpose of the Pathmaker routine is to plot an optimal path in the initial stages of the encounter, from time of launch until the target is detectable within the FOV. Once the target is within the FOV, tracking will be transferred to a separate Cursor-on-Target (CoT) tracking function.

This algorithm allows the user to identify the target within the FOV and establish tracking through a semi-autonomous interface. For more information on Cursor-on-Target tracking, refer to Lt. Josh Crouse's thesis work (Crouse, 2007), as well as the future thesis work of Lt. Troy Vantrease. Lt. Vantrease is conducting a thesis on integrating Lt. Crouse's CoT techniques into the current MAV platform.

One other point to keep in mind is the fact that the Pathmaker algorithm is optimized for a singe point, and has no recognition of whether the target may be in any other part of the FOV. It may be instructive to take the example given in Figure 25 and overlay a standard camera FOV onto it as seen in Figure 26. Figure 26 depicts a camera FOV and elevation angle of 60 and 45 degrees respectively, from an altitude of 120 meters. The three FOV snapshots visible are taken just prior to making the initial turn to intercept, just after the 1st turn to intercept, and finally at 30 seconds into the entire flight path. This last point also correlates to the first turning point for the moving target.

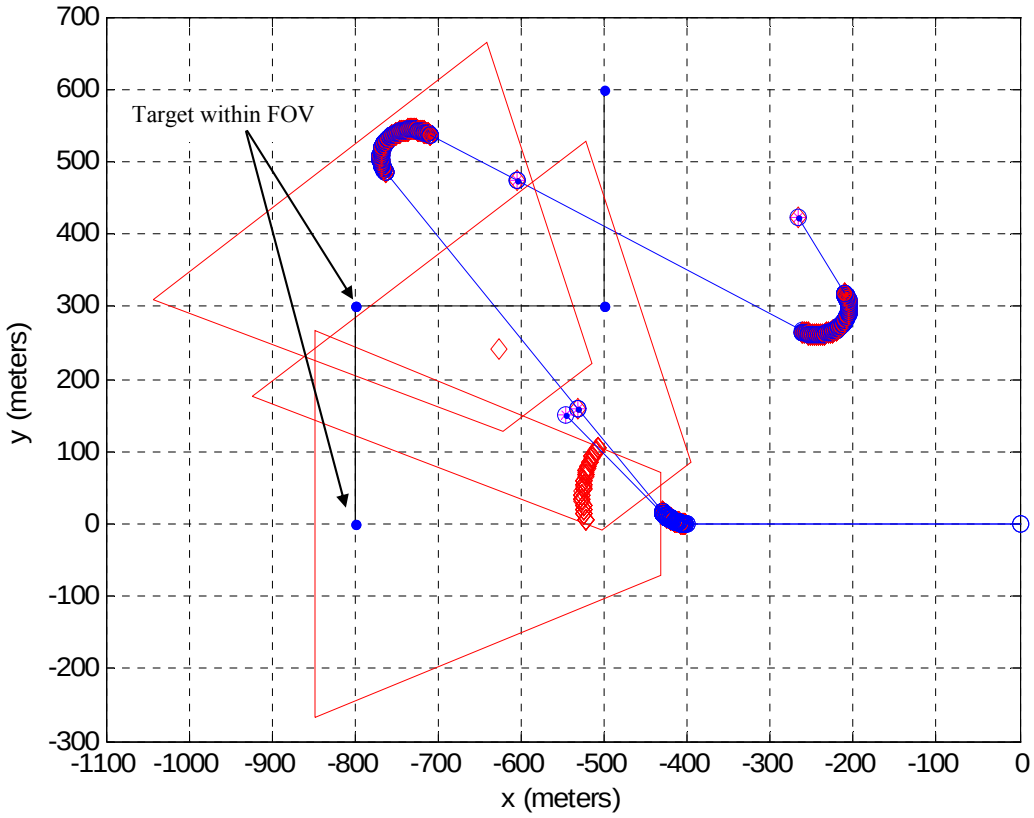


Figure 26. High lag time solution with FOV overlay

Figure 26 demonstrates that the target was actually within the detectable FOV from the time just prior to the turn to intercept, all the way until the target changed course. However, because Pathmaker optimizes for a single point along the camera's bore sight, it was not considered to be legitimate solution. In a real world scenario, once the target was detected within the FOV, the user would have transferred to a CoT tracking interface to maintain the target in the FOV. In this respect, Pathmaker would have been successful in guiding the UAV to a point of interception as it was intended to do.

It should be noted that this information lag problem is endemic to any optimization routine, and not solely the Pathmaker algorithm. Zollars' dynamic optimization routine, for example, would have to cope with the same issue of maintaining up-to-date information on the current location of the target. Inherent in this argument is the idea that you have, by some method or another, received current information on the target. In the Conclusions and Recommendations chapter, we will discuss some ways that can minimize the time between change in target direction and execution of a new flight path solution will be discussed.

3.2.8 Summary

A heuristic, iterative algorithm was developed that significantly reduced the calculation time while also incorporating features to compensate for computation time lag. Results were presented that demonstrated promising capability of tracking real world stop-and-go targets as well as variable winds. Results were also presented to prove that the Pathmaker solution provides superior results when compared to other computationally intensive optimal path techniques, given real-world time lag constraints. The Pathmaker routine also proved to be incredibly robust and resistant to computational crashes with predictable execution times, a trait that has plagued some of the earlier dynamic optimization routines. To fully prove the viability of the Pathmaker approach, flight testing will be conducted. This work will be presented in Chapter 4. Actual Matlab code is contained in Appendix B.

4. Flight Test Results and Analysis

4.1 Overview

A full set of flight testing was not completed due to inclement winter weather and time constraints. A limited series of flight tests were conducted in November of 2007 however . This chapter will discuss these flight tests and any valuable data that was obtained.

The following flight tests were accomplished rather early in thesis development process. Because a new airframe and autopilot system were being used that had not been validated in the past, the first flight tests were spent setting up the autopilot and verifying that the various hardware systems were working properly. The majority of the achieved flight testing involved this validation process, with a limited set of Pathmaker test cases being accomplished.

4.2 Validating the Sig Rascal with Kestrel Autopilot

As mentioned in Chapter 2, AFIT has used the Sig Rascal extensively in the past with the Piccolo autopilot system, but it was necessary to validate the Sig Rascal airframe using the Kestrel autopilot. Because of recent updated interpretations of FAA law, it was no longer possible to conduct flight testing at Wright-Patterson AFB, so flight testing was conducted at Camp Atterbury, IN.

Procerus provides a detailed process for configuring the autopilot for the first flights in their "Installation and Configuration Guide" (Kestrel, 2008). A brief description of this Procerus' autopilot configuration and first flight tests are listed below:

1. Prior to 1st Flight

a. Servos are configured

b. Autopilot settings and gains are configured with default parameters.

2. 1st Flight

    a. Verify proper operation of critical autopilot systems such as the autopilot sensor suite, the communication link, and the GPS receiver.

    b. Trim the aircraft.

    c. Find the trim airspeed, trim throttle, and trim angle of attack.

3. 2nd Flight

    a. Tune the rate damping servo loops. The rate damping loops damp the aircraft rotation around the pitch, roll, and yaw axis if the aircraft has a rudder.

4. 3rd Flight

    a. Tune the attitude, altitude and airspeed autopilot loops.

5. 4th Flight

    a. Verify waypoint navigation and loiter work correctly.

These first 4 flights were conduct following Kestrel's procedural outline. Typical flight test procedure was to take off under pilot command and ascend at an airspeed between 25-30 m/s to an altitude of 200 meters. Once at altitude, the pilot would conduct a series of in-flight checks, and once he was comfortable that the aircraft was operating correctly, the Kestrel flight test regimen was conducted.

Flight test results for this validation process were successful with no complications. The initial values that were given for the autopilot were accurate for the

airframe and very little modification had to be made. With the autopilot configured, the Sig Rascal proved to be a reliable platform and performed well in the given conditions.

In conclusion, the Sig Rascal was validated and verified for use with the Kestrel autopilot system. This UAS can now confidently be used as a test platform for any future UAV research at AFIT.

4.3 Camera Validation

A major component required for the success of any Pathmaker validation and target tracking was the validation of the camera system. Initial tests with the camera proved successful. Figure 27 is an example of a still image taken from the aircraft at an altitude of approximately 200 meters. The light brown rectangles toward the center of the image are Army Humvees (circled in red). The elevation angle of the camera was approximately 40 degrees. These vehicle sized targets were readily detectable within the camera FOV at this altitude.

Figure 27. UAV camera showing Humvees at 200m Altitude

Unfortunately, additional camera testing proved problematic after the 1st flight. Persistent white banding began to occur within the camera FOV as shown in Figure 28. This banding occurred irrespective of aircraft attitude and orientation. Originally it was thought that this had to do with antenna interference, however a different antenna setup was tried and did not alleviate the problem.

Figure 28. White banding anomaly seen in camera FOV

Attempts at isolating the issue were unsuccessful. Furthermore, attempts at recreating the banding on the ground in the lab were unsuccessful. There is a possibility that there is an electromagnetic interference between the video transmission hardware and the portable generator used to power the ground station. This has not been confirmed and should be investigated in the future as a potential problem source.

Despite the white banding that was seen, some good reference data was accumulated. From an altitude of 200 meters, vehicle sized targets were readily detectable within the camera FOV. The cameras were set up with 64 degree FOV lenses. Human sized targets were minimally detectable within the FOV at this altitude. In Figure

29 a bicyclist can be seen on the road (circled in red). While hardly visible within the still image, the person was easier to see within the video feed. This would relate to human physiology and the eyes' acute sensitivity to peripheral motion. Figure 29 does prove however that a human is detectable from this level and will enter future optimal tracking altitude considerations.

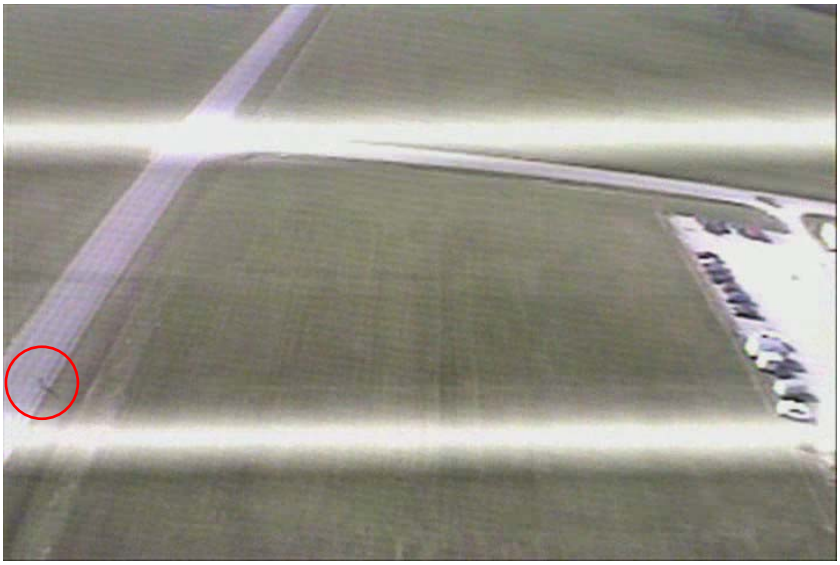


Figure 29. Camera FOV with bicycler circled in red

Camera work was discontinued in the later flights because some of the camera hardware shook loose. As a result, no camera support was available to validate the Pathmaker routine. Since the November test flights the cameras have been shock mounted and enclosed in a plexiglas bubble to alleviate any windstream disturbances. This new camera setup was not tested before the conclusion of this thesis.

4.4 Pathmaker algorithm flight testing

A limited series of tests of the Pathmaker algorithm were conducted. With the time remaining after the initial Kestrel flight tests were completed, time was spent

validating the UAV camera video feed, FTC controller interface and the Pathmaker algorithm.

To confirm that Pathmaker was working properly, members of the group picked a target located away from the ground station unit. Using a portable GPS, they recorded the lat/long position of the target. These coordinates were entered into the FTC which executes Pathmaker. Upon execution, Pathmaker generated a series of 2 waypoints that directed the UAV to the target.

Because video feed was not available to confirm the location of the targets within the FOV, correct target acquisition was inferred from the Virtual Cockpit UAV tracking screen. Figure 30 shows a screen capture of a Pathmaker solution.

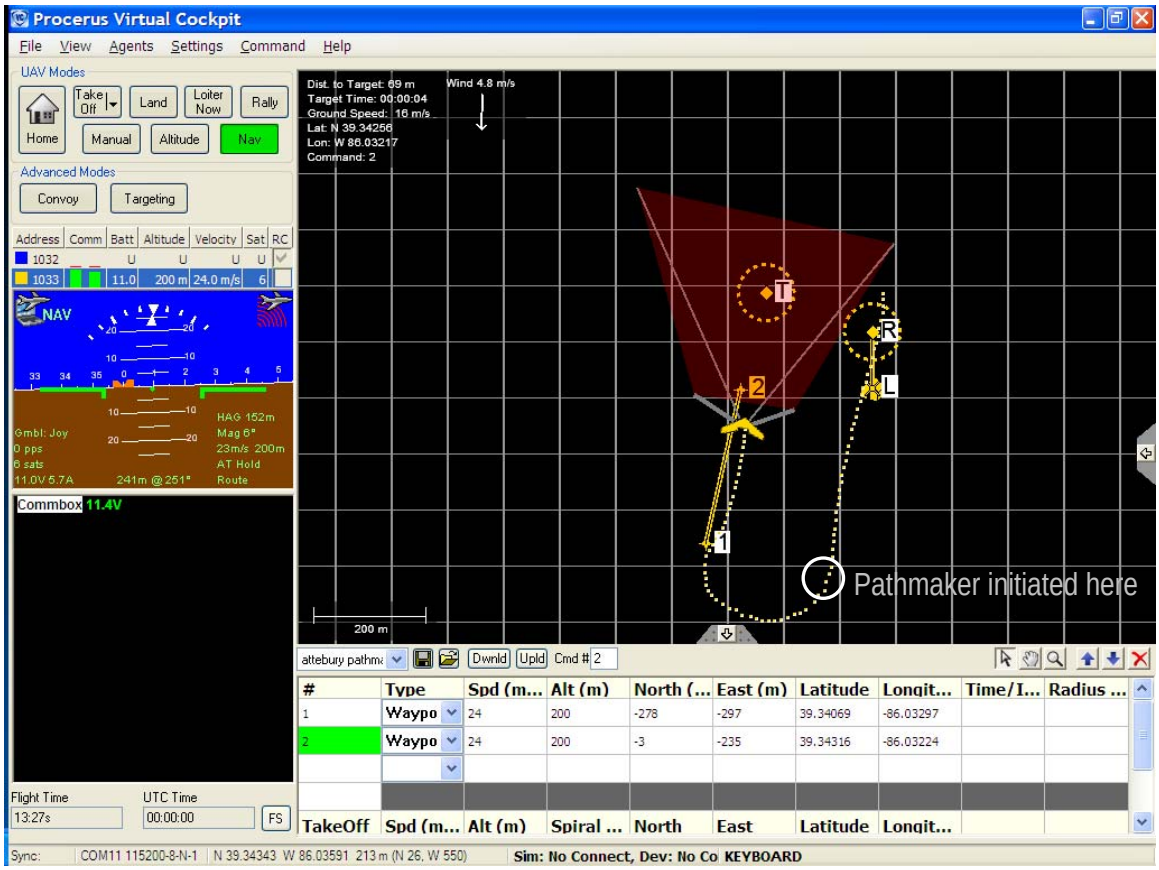


Figure 30. 2-Point Pathmaker Solution with Camera FOV shown in red

The test was started with the UAV flying in a southerly direction away from the target. The known position of the target was marked by dashed circle "T" waypoint to help determine the accuracy of the Pathmaker solution. As can be seen in Figure 30, Pathmaker successfully created as series of waypoints that captured the target in the FOV, which is represented by the red translucent trapezoid.

A similar test was conducted to test the ability of Pathmaker to predict the location of a moving target. Using the same known target, a moving speed and direction were entered into the FTC, and Pathmaker was executed. Figure 31 is a screenshot of the UAV intercepting a target moving due west from the original target location at 5 m/s.

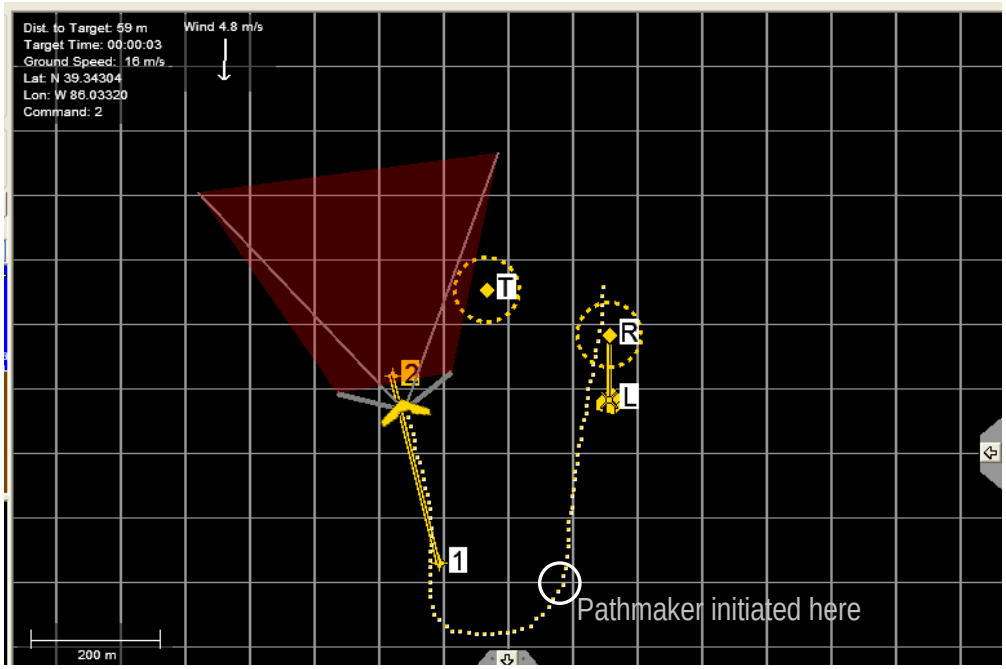


Figure 31. Pathmaker Solution for simulated moving target

Because we did not actually have a moving target available, this test was conducted simply to verify if the sense of Pathmaker's predictions were correct. For example, if the target was moving west, then the final waypoint should be some position west of the

original position. The flight time from Pathmaker execution to waypoint 2 was approximately 30 seconds. With a target moving at 5 m/s, the intercept point should be approximately 150 meters west of the starting target location. Figure 31 indicates that Pathmaker correctly estimated the location of the target.

Unfortunately no other flight testing was possible. The first round of flight testing was meant only to validate the Sig Rascal/Kestrel system as a UAV platform, as well as test basic FTC systems and algorithm execution. A second in-depth validation of Pathmaker, using live moving targets was never accomplished due to time and weather constraints. Recommendations for future flight testing are included in Chapter 4.

5. Conclusion and Recommendations

5.1 Conclusions

The goal of this research was to develop a wind corrected, moving target UAV flight planning algorithm that could produce accurate target intercepts in real time. An algorithm was successfully created that could effectively navigate a UAV in real-time and determine an optimal flight path starting at an initial position and heading and ending with a desired viewing angle of the target. The solution technique developed used a Dubins-style flight path model optimized through a heuristic iterative approach. This approach proved to be much less time consuming to calculate and produced equivalent results as other more computationally intensive dynamic optimization routines. Specifically, the following was accomplished:

- A method was created to compensate for computational and communication relay time lags that extrapolated the future position of the aircraft at point of command execution.
- Methods were discussed of how to create an effective auto-updating wind corrected flight path using the developed algorithms.
- The Sig Rascal was validated and verified for use with the Kestrel autopilot.

Flight testing of the actual algorithm was limited due to weather restrictions. However the limited flight testing that was completed indicate that the Pathmaker algorithm effectively intercept the target. HIL testing also indicate that the developed algorithms successfully guide the UAV to the target. Additional flight testing is needed to validate the system, however.

5.2 Recommendations for Future Research

5.2.1 More flight testing

Additional flight testing is required to fully validated the Pathmaker algorithms as an effective optimal flight path tool. Specific areas of testing that are required are as follows:

1. Verify ability to place UAV sensor on stationary target
2. Verify ability to place UAV sensor on moving target
3. Verify ability to adequately track a stop-and-go moving target
4. Verify ability to adequately track an unpredictable (nonlinear) moving target

A flight test procedure document was created that was intended to be used had flight testing occurred. It has been included in Appendix C, and should provide a good starting point for future groups interested in developing a flight test program.

5.2.2 AVDS HIL sim

Due to the nature of the education cycle, it has become increasingly apparent that a robust HIL system is required that can incorporate variable camera angles and moving targets. Graduate students generally do not begin serious work on their thesis until the fall.

For this reason, it is advisable that a future groups focus on creating a HIL system that can more effectively test the developing capabilities of the UAV. Two key components are required: the ability to change to displayed FOV to mimic current camera settings, and the ability to incorporate a moving target. The current HIL software

(Aviones) packaged with the Kestrel autopilot does not allow for moving target visualization or variable camera angles.

One piece of software that is already available to AFIT which may provide a solution to this problem is AVDS. Lt Crouse (Crouse, 2007) used AVDS extensively in his thesis work, to include tracking moving targets. AVDS is a piece of simulation software that allows the user to change the FOV to various camera angles. For more information on AVDS, refer to Lt Crouse's thesis, or the AVDS website (AVDS, 2008).

5.2.3 Aircraft Heading Verification

During the course of this thesis research, it was assumed that the UAV magnetic heading (in contrast to the ground track heading) is known and accurate. Both values are contained within the telemetry feed from the aircraft. The ground track heading is readily derived from the vector path of the UAV, while the physical heading, or bearing of the UAV is not as obvious. In fact, it is not entirely obvious how the Kestrel autopilot is deriving this value. A future effort should be made to verify that the UAV bearing data is accurate, and an effort should be made to characterize the effect of

5.2.4 Implement Auto-wind correcting into FTC

One takeaway from the Pathmaker analysis was that it is possible to make varying wind correction an automatic loop within the system architecture. Below is a plausible solution which could be implemented in the future:

1. Record the wind speed data from the autopilot telemetry output.
2. Create a tolerance range for the wind.

a. Example: if the wind is within 1 knot, and the wind direction is within 5 degrees of the last Pathmaker execution, do nothing.
b. Otherwise, execute Pathmaker

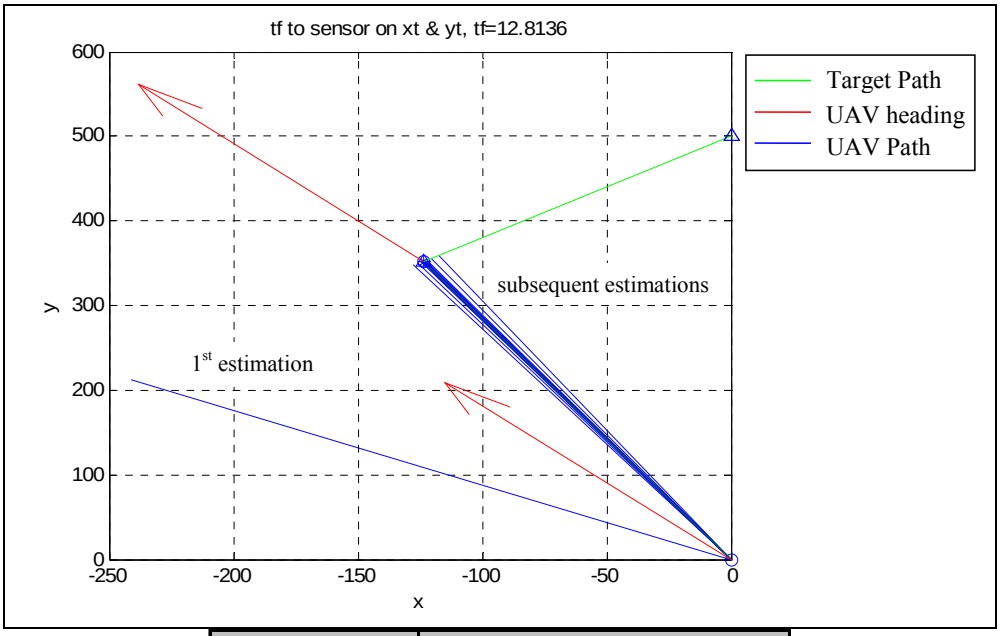
3. Create flags within the system architecture that will execute Pathmaker given the stated tolerances.

Because the wind data is coming from the UAV, this process can be implemented to operate autonomously without any user input and has the potential of improving path optimization over long flight distances.

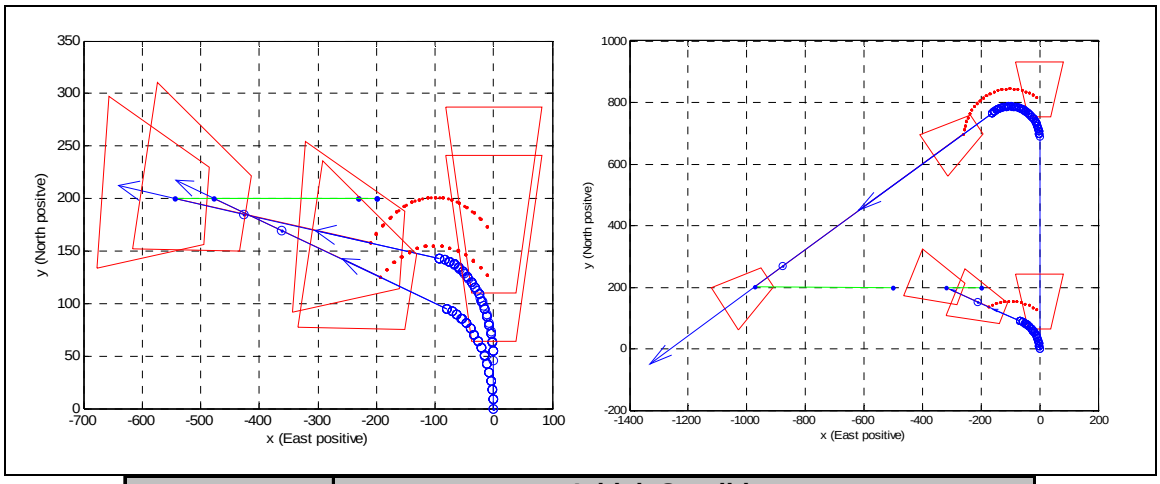
In summary, a viable path planning solution was derived that shows great potential for future work in real world UAV and MAV flight path generation. While initial tests appear to confirm the viability of the Pathmaker algorithm, additional flight testing is necessary to validate the algorithm in real world scenarios.
inaccurate UAV bearing data on flight path calculation and sensor placement.

Appendix A. Additional Pathmaker Validation Information

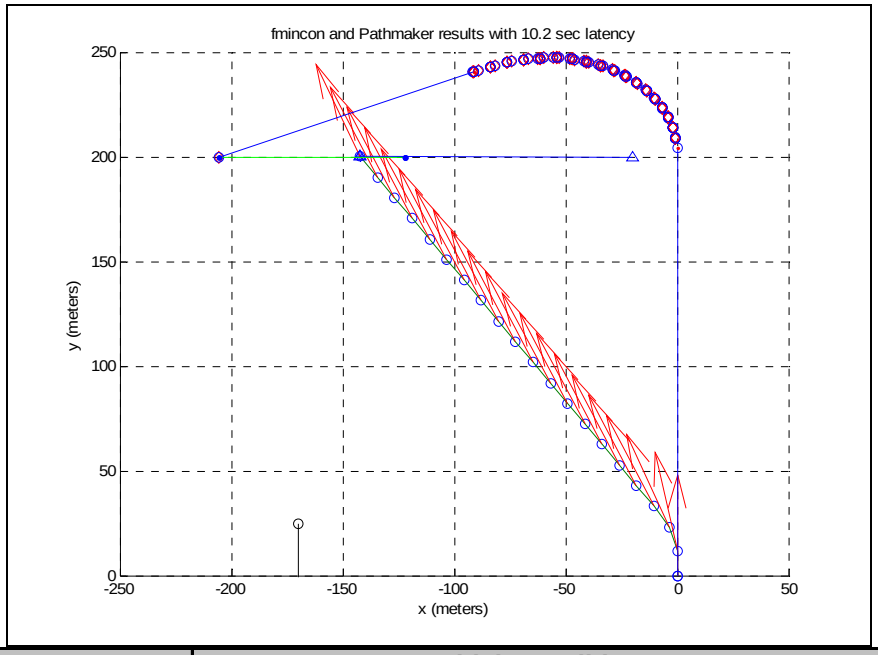
The following are a series of graphs and tables constructed during the Pathmaker analytical process. The graphs depict the results of Pathmaker algorithm executions and the tables attached below each graph included the pertinent information for each run.



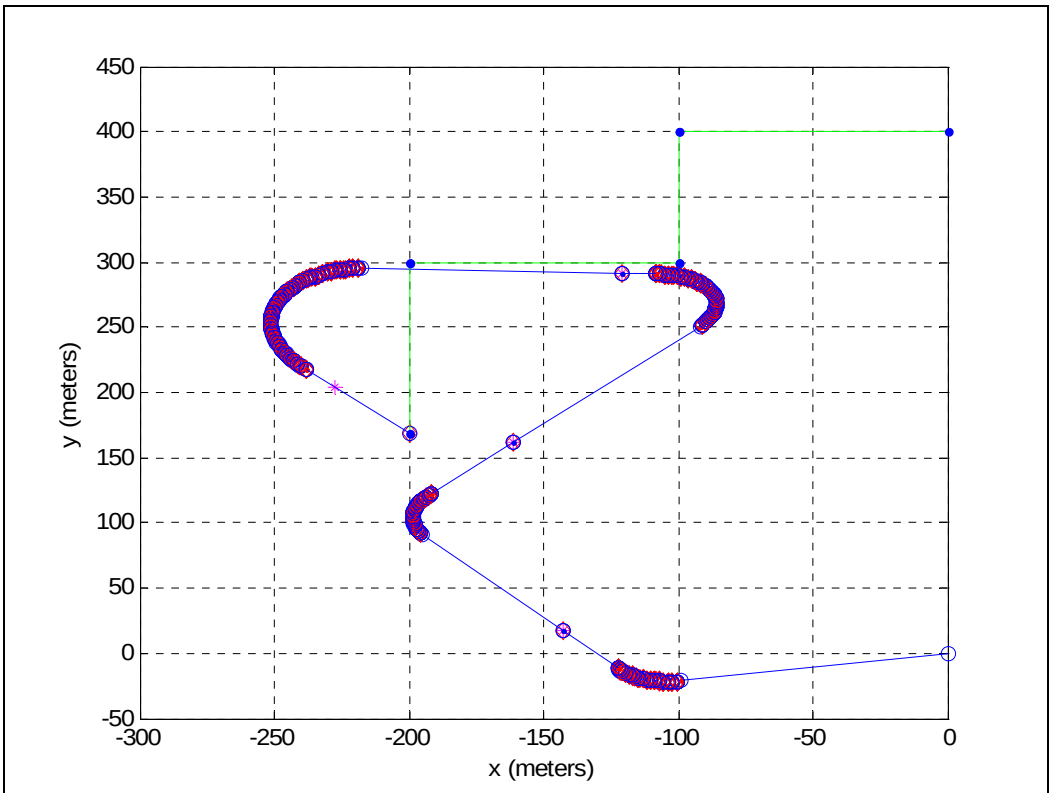
	Initial Conditions		
	Target	UAV	Wind
Initial X (m)	0	0	
Initial Y (m)	500	0	
Speed (m/s)	15	20	10
Heading (deg)	220	0	180



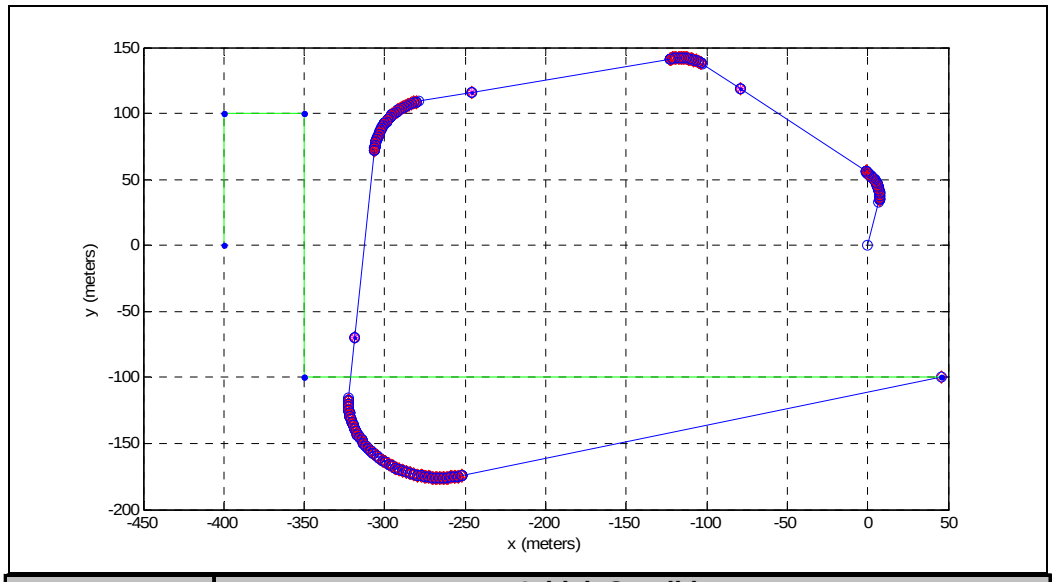
	Initial Conditions				
	Target	UAV	Wind	Camera	
Initial X (m)	-200	0		Elevation	45
Initial Y (m)	200	0		Azimuth	0
Speed (m/s)	10	25	10	FOV	30
Heading (deg)	270	0	180	Altitude (m)	100
Lag Time (s)		varied			



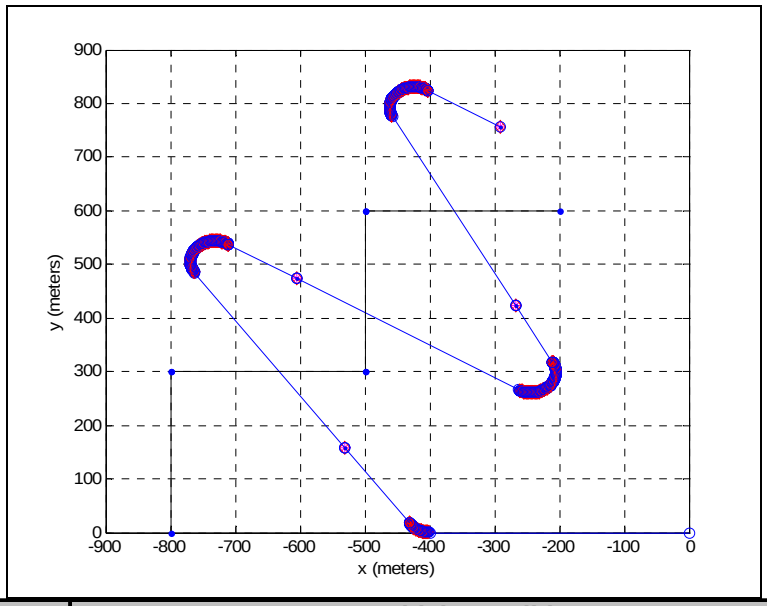
	Initial Conditions				
	Target	UAV	Wind	Camera	
Initial X (m)	0	0		Elevation	90
Initial Y (m)	200	0		Azimuth	0
Speed (m/s)	10	30	10	FOV	0
Heading (deg)	270	0	0	Altitude (m)	100
Lag Time (s)		10.2			



	Initial Conditions					
	UAV	Wind	Target:	leg 2	leg 3	leg 4
Initial X (m)	0		0	-100	-100	-200
Initial Y (m)	0		400	400	300	300
Speed (m/s)	20	5	10	10	10	10
Heading (deg)	0	315	270	180	270	180
Lag Time (s)	6					



	Initial Conditions					
	UAV	Wind	Target:	leg 2	leg 3	leg 4
Initial X (m)	0		-400	-400	-350	-350
Initial Y (m)	0		0	100	100	-100
Speed (m/s)	20	5	10	5	20	19.5
Heading (deg)	0	315	0	90	180	90
Lag Time (s)	2					



	Initial Conditions					
	UAV	Wind	Target:	leg 2	leg 3	leg 4
Initial X (m)	0		-800	300	300	600
Initial Y (m)	0		0	-800	-500	-500
Speed (m/s)	20	0	10	10	10	10
Heading (deg)	270	0	0	90	0	90
Lag Time (s)	20					

Appendix B. Matlab Code

B.1 Pathmaker Routines

```
function count = startturnpath(in)
%Pathmaker Initialization file
%To be used as input to pathmaker.m
%Nate Terning
clc
clear all; %clf
% tic
if(nargin==0)
d2r = pi/180;
N=2;
MaxIts=[200];  % use default
MaxFunEvals=500;    % vehicle IC
thf = 90; %final intercept angle (absolute),
constrain_theta = 0; %thf on:1 off:0
d = 200; %lookout Distance, out left side of UAV(m)

%*****************  DATA I NEED PASSED IN
*******************************
% targetDat
x_t = -100;             %Target relative x position,m
y_t = 1000;              %Target relative y position,m
hdg_t = 0*d2r;         %target heading (currently in GPS rad coorinates)
vel_t = 10;              %target velocity, fps (15m/s ~= 30kts)
alt_t = 0;              %target altitude

%uavDat
vel_u = 23;             %UAV true airspeed, m/s
x_u = 0;                %UAV x position
y_u = 0;                %UAV y position
roll_u = 0;             %UAV roll orientation (for camera look angle)
pitch_u = 0;            %UAV pitch angle
hdg_u =  0*d2r;         %UAV heading (currently in GPS rad coords)
alt_u = 50;              %UAV altitude 61m ~ 200ft

%wind = [speed hdg]
v_wind = 5;             %wind speed, m/s
hdg_wind = 315*d2r;     %wind blowing FROM direction (GPS rad coords)

%sensorDat = [camElevation camAngle camFOV];
camElevation = 90*d2r;      %camera look-down angle
camAngle = 0*d2r;          %0: out nose; 90:  LHS; -90: RHS
camFOV = 0*d2r;            %Field of View of camera
%
*****************************************************************
**

runData = [N,MaxIts,MaxFunEvals]';
```

```
targetDat = [x_t y_t hdg_t vel_t alt_t,thf, d,constrain_theta]';
uavDat = [x_u y_u roll_u pitch_u hdg_u vel_u alt_u]';
wind = [v_wind hdg_wind]';
sensorDat = [camElevation camAngle camFOV]';

%PACK AND PLAY
[x_out, y_out, th_out] = pathmaker(runData', targetDat',uavDat',wind',
sensorDat');
disp('      X          Y       Theta')
disp([x_out y_out th_out./d2r])

end %nargin = 0
% toc
return %startturnpath
```

```
function [x_out, y_out, th_out] = pathmaker(runData,
targetDat,uavDat,wind, sensorDat)
%Pathmaker Version Date 05 Nov 2007
%   Removed input_out return array
%   fixed a problem with hdg_uw (th_out) being measure incorrectly
%   Create argument that takes into account turning radius
%   Build in latency time into function (ie time it takes to transfer
soln
%     to UAV) 27Nov
%   Create code that will allow sensor footprint offset 30 Nov
%   Create visualization for camera(sensor) footprint
%     NOTE:  currently assumes camera no bank angle 03Dec2007
%   turn optimization was re-written to aid turns > 180deg 07Dec2007
%   This version will look at including pitch-down info for terminal
phase 11Dec


% clc;
d2r = pi/180;

%UNPACK ALL THE INPUT DATA:
%runData
N = runData(1);
MaxIts = runData(2);
MaxFunEvals = runData(3);


%UAV Dat
    x_u = uavDat(1); x_u1 = x_u;
    y_u = uavDat(2); y_u1 = y_u;
    roll_u = uavDat(3);
    pitch_u = uavDat(4);
    hdg_u =  (pi/2 - uavDat(5));         %UAV heading (currently in d2r
coords)
        if hdg_u < 0, hdg_u = hdg_u + 360*d2r; end  %Aids conversion to
magnetic heading
    vel_u = uavDat(6);
    alt_u= uavDat(7);

%targetDat = [x_t y_t hdg_t vel_t alt_t]'
    x_t = targetDat(1);  x_t1 = x_t;
    y_t = targetDat(2);  y_t1 = y_t;
    hdg_t = (pi/2 - targetDat(3)); %target heading (currently in rad
coorinates)
        if hdg_t < 0, hdg_t = hdg_t + 360*d2r; end  %Aids conversion to
magnetic heading
    vel_t = targetDat(4);
    alt_t= targetDat(5);

%Wind conditions
    W= wind(1);
    th_w= (pi/2 - wind(2));
    Wx=W*cos(th_w);
    Wy=W*sin(th_w);
```

```
%Camera Data
    camElevation = sensorDat(1); %camera look-down angle
    camAngle = sensorDat(2); %0: out nose; 90:  LHS; -90: RHS
    camFOV = sensorDat(3); %Field of View of camera
    FOVtype = 2;
 %The following exception is made for current FTC process that requires
 %final waypoint to always be on target
    camElevation = 90*d2r; %camera look-down angle
    camAngle = 0; %0: out nose; 90:  LHS; -90: RHS
    camFOV = 0; %Field of View of camera

    if FOVtype ==1,
        %This takes the bisector of the max and min field distances for
        %sensor target
        fieldlimit = 900; %max horizon distance you want to include in
footprint
        %NOTE: Also need to change in createFOV function!
        flines =
[alt_u/tan(camElevation+camFOV/2),alt_u/tan(camElevation-camFOV/2)];
        if abs(flines(2))>fieldlimit, flines(2)=fieldlimit;end
        d = sum(flines)/2;
    else
        %Because it may be more optimal to put the target at the
boresight of
        %the camera the following is developed as an alternative:
        d = alt_u/tan(camElevation); %distance looking out through
center of camAngle
    end

%     createFOV(x_u, y_u, hdg_u, alt_u, camAngle, camElevation, camFOV,
0);


%Create Latency path (path UAV follows before commands are sent)
latency = 2; %seconds NOTE: this can be changed in future if needed
%    latency = 2 + .0001*norm([x_t-x_u, y_t-y_u]);
   vel_uwx = vel_u*cos(hdg_u) - Wx;
   vel_uwy = vel_u*sin(hdg_u) - Wy;
   x_u = x_u + vel_uwx*latency;
   y_u = y_u + vel_uwy*latency;
   x_t = x_t + vel_t*(cos(hdg_t))*latency; %expected pos of target
   y_t = y_t + vel_t*(sin(hdg_t))*latency;
   [x_s,y_s] = getSensorLoc(x_u, y_u, hdg_u, camAngle, d,0,'sensor');
   %plot([x_u1 x_u],[y_u1,y_u], 'bo',[x_u1 x_u],[y_u1,y_u], 'b'); hold
on
   createFOV(x_u, y_u, hdg_u, alt_u, camAngle, camElevation, camFOV,
0);

%CREATE INITIAL OPTIMIZATION
[x_us y_us x_t_e y_t_e hdg_uw, tti count] = CourseOpt(vel_u, vel_t,
hdg_t, Wx, Wy, x_t, y_t, x_u, y_u,camAngle,d, hdg_u);
%x_us denotes location of UAV necessary to put sensor on target
```

```
%The following will try to incorporate pitch-down angle for terminal
phase.
% phasetype = 'loiter  ';
% phasetype = 'terminal';
phasetype = 'other   ';
    if phasetype == 'terminal',
        range_t = sqrt((x_u-x_t_e)^2+(y_u-y_t_e)^2);
        camElevation=camElevation+atan2(alt_u,range_t);
        d = alt_u/tan(camElevation);
    end
    if phasetype == 'loiter  ';
        range_t = sqrt((x_u-x_t_e)^2+(y_u-y_t_e)^2);
        camElevation=camElevation+15*d2r; %Currently just assuming 15
deg loiter bank angle
        d = alt_u/tan(camElevation);
    end

%Now, segment into N waypoints
x_g = linspace(x_u,x_us,N);
y_g = linspace(y_u,y_us,N);
hdg_uw = hdg_uw*ones(1,N);
% plot(x_g,y_g, '*', x_g,y_g); grid on; hold on %plot of initial guess
% plot(x_g,y_g, 'bo'); grid on; hold on %plot of initial guess
sw = 0; swfix = 0; %switch to determine if UAV has switched from RH to
LH turn
mtr = 15*d2r; %max turn rate in rad/sec



turnside = 'lh';
%Determine both left and right paths:
[x_ul,y_ul,x_usl,y_usl,hdg_uwl,ttcl,ttil,x_t_e,y_t_e] =...

turnpath(mtr,hdg_uw,Wx,Wy,hdg_u,vel_u,x_u,y_u,hdg_t,x_t,y_t,vel_t,camAn
gle,d,turnside,0);

turnside = 'rh';
[x_u2r,y_u2r,x_usr,y_usr,hdg_uwr,ttcr,ttir,x_t_e,y_t_e] = ...

turnpath(mtr,hdg_uw,Wx,Wy,hdg_u,vel_u,x_u,y_u,hdg_t,x_t,y_t,vel_t,camAn
gle,d,turnside,0);

if (ttcl+ttil) > (ttcr+ttir), turnside = 'rh';else turnside = 'lh';end
%run again to print
[x_u2,y_u2,x_us,y_us,hdg_uw,ttc,tti,x_t_e,y_t_e] = ...

turnpath(mtr,hdg_uw,Wx,Wy,hdg_u,vel_u,x_u,y_u,hdg_t,x_t,y_t,vel_t,camAn
gle,d,turnside,1);

%The following will mark a certain time point in the flight path
%(use to determine turning pts for stop/go targets)
y10 = y_u2 + (y_us - y_u2)*(10 - latency - ttc)/(tti-ttc);
x10 = x_u2 + (x_us - x_u2)*(10 - latency - ttc)/(tti-ttc);
y_t10 = y_t1 + (y_t_e - y_t1)*(10)/(tti + ttc + latency);
```

```
x_t10 = x_t1 + (x_t_e - x_t1)*(10)/(tti + ttc + latency);
% disp(['   x10      y10'])
% disp([x10,y10])
% plot(x10,y10,'m*')

%Now, segment into N waypoints
x_g = linspace(x_u2,x_us,N);
y_g = linspace(y_u2,y_us,N);
% x_g = linspace(x_u2,x10,N);  %USE FOR STO-GO ONLY!
% y_g = linspace(y_u2,y10,N);  %USE FOR STO-GO ONLY!
hdg_uw = hdg_uw*ones(1,N);
[x_s,y_s] = getSensorLoc(x_g, y_g, hdg_uw, camAngle, d,1,'sensor');
% disp('     dx_s        dy_s');disp([x_s(N)-x_t_e;y_s(N)-y_t_e]');
for i = 1:N
    createFOV(x_g(i), y_g(i), hdg_uw(i), alt_u, camAngle, camElevation,
camFOV, 1);
end

th_out = 90*d2r - hdg_uw;
if th_out < 0, th_out = th_out + 360*d2r; end  %converts to positve
magnetic heading
x_out = x_g';
y_out = y_g';
th_out = th_out';

%***************  PLOTTING RESULTS  ****************************
%*************************************************************
plotcourse = 1;
if plotcourse == 1
    %For quiver function
    Vx_g=[.2*vel_u*cos(hdg_uw)];
    Vy_g=[.2*vel_u*sin(hdg_uw)];

    %figure(1); %plot wayponts
    %plot(x_g,y_g,'bo',x_g,y_g, 'b'); grid on; hold on; %axis([-50 200
-50 200]);
    %plot(x_g(N),y_g(N),'b')
%     quiver(x_g,y_g,Vx_g,Vy_g,'b')
%     xlabel('x (meters)');
%     ylabel('y (meters)')
%      title(['tf to sensor on xt & yt, tf=',num2str(tti+ttc+latency)])

    % %***********  VEHICLE GROUND PLOT  *********************
    x_vh=[x_t1, x_t_e];
    y_vh=[y_t1, y_t_e];
    %plot(x_vh,y_vh,'g',x_t1,y_t1,'b.',x_t_e,y_t_e,'b.')
%     x_vh=[x_t1, x_t10]; %FOR STOP AND GO
%     y_vh=[y_t1, y_t10]; %FOR STOP AND GO
%     plot(x_vh,y_vh,'g',x_t1,y_t1,'b.',x_t10,y_t10,'b.')


end %plot
count;
return %Pathfinder
```

```
function [x_u2 y_u2 x_t_e y_t_e hdg_uw tti count] = CourseOpt(vel_u,
vel_t, hdg_t, Wx, Wy, x_t, y_t, x_u, y_u,camAngle,d, hdg_u)
%CourseOpt is the subroutine that calculates an optimized flight path
%It uses a hueristic approach to iterating the following:
% 1.    Determine UAV sensor location
% 2.    Calculate the range between sensor and target
% 3.    Calculate time-of-flight (TOF) required to intercept target at
current location.
% 4.    Calculate expected location of target given current time-of-
flight.
% 5.    Estimate heading required to compensate for wind, given wind
speed and TOF
% 6.    Calculate necessary UAV final location, given wind correction
and sensor footprint location
% 7.    Estimate actual ground speed
% 8.    Re-calculate range given updated expected location of target
% 9.    Iterate process until range estimations from steps (2) and (8)
are equivalent.

d2r = pi/180;
%Initialize Variables
x_t_e = x_t; y_t_e = y_t; hdg_uw=hdg_u;
Range2 = 0; Range = 10; count = 0;
xrel = x_t_e - x_u;   yrel = y_t_e - y_u;
    %converge onto actual range by iterating
    while abs(Range2 - Range) > .01 & count<200
        [x_s,y_s] = getSensorLoc(x_u, y_u, hdg_uw, camAngle,
d,0,'sensor');
        Range = sqrt(xrel^2 + yrel^2);
        tti = Range/vel_u;  %rough wag on minimal time-to-impact
        x_t_e = x_t + vel_t*(cos(hdg_t))*tti; %expected pos of target
        y_t_e = y_t + vel_t*(sin(hdg_t))*tti;
        xrel = x_t_e  - x_s + Wx*tti; %incorporates wind correction
        yrel = y_t_e  - y_s + Wy*tti;
        hdg_uw = atan2(yrel,xrel); %current heading estimation
        %position of UAV given sensor location:
        [x_u2,y_u2] = getSensorLoc(x_t_e, y_t_e, hdg_uw, (camAngle-
180*d2r), d,0,'uav   ');
        vel_uwx = vel_u*cos(hdg_uw) + Wx;
        vel_uwy = vel_u*sin(hdg_uw) + Wy;
        vel_uw = norm([vel_uwx, vel_uwy]); %ground speed
        Range2 = sqrt((x_u2-x_u)^2 + (y_u2-y_u)^2); %norm([xrel yrel
hrel]);
%         Range2 = sqrt(xrel^2 + yrel^2);
        count = count+1;
    end
return %CourseOpt


function  [x,y] = ellipse(a,b,phi,x0,y0,n)
% ELLIPSE  Plotting ellipse.
%   ELLIPSE(A,B,PHI,X0,Y0,N)  Plots ellipse with
%   semiaxes A, B, rotated by the angle PHI,
%   with origin at X0, Y0 and consisting of N points (default 100).
%   [X,Y] = ELLIPSE(...) Instead of plotting returns coords of ellipse.
```

```
%  Copyright (c) 1995  by Kirill K. Pankratov
%   kirill@plume.mit.edu
%   03/21/95
n_dflt = 100;   % Default for number of points
 % Handle input .................
if nargin < 6, n = n_dflt; end
if nargin < 5, y0 = 0; end
if nargin < 4, x0 = 0; end
if nargin < 3, phi = 0; end
if nargin < 2, b = 1; end
if nargin < 1, a = 1; end
th = linspace(0,2*pi,n+1);
x = a*cos(th);
y = b*sin(th);
c = cos(phi);
s = sin(phi);
th = x*c-y*s+x0;
y = x*s+y*c+y0;
x = th;
if nargout==0,
    %plot(x,y);
end
```

```
function [x_s,y_s] = getSensorLoc(x_u, y_u, hdg_u, camAngle,
d,plotflag,icon)
%getSensorLoc will determine the location of the sensor given the UAV
%location & camera info.  Note that the sensor location is modeled as a
point.
%The process can also be done in reverse to determine the UAV location
from
%a known sensor point.
x_s = x_u + d*cos(camAngle + hdg_u);
y_s = y_u + d*sin(camAngle + hdg_u);
if plotflag==1
    if icon == 'sensor',icon = 'rd';
    elseif icon == 'uav   ',icon = 'bo';end
    %plot(x_s,y_s, icon); hold on;
end
return  %getsensorLoc
```

```
function [flines] = createFOV(x_u, y_u, hdg_u, alt_u, camAngle,
camElevation, camFOV, plotflag)
%createFOV provides trapezoidal representation of UAV sensor FOV
%currently does not account for bank angle

d2r = pi/180;
fieldlimit = 90000; %max horizon distance you want to include in
footprint
%NOTE: Also need to change in createFOV function!
flines = [alt_u/tan(camElevation+camFOV/2),alt_u/tan(camElevation-
camFOV/2)];
        if abs(flines(2))>fieldlimit, flines(2)=fieldlimit;end
range =  [norm([flines(1),alt_u]), norm([flines(2),alt_u])];
fRH  = [range(1)*tan(camFOV/2), range(2)*tan(camFOV/2)];
fLH  = [-range(1)*tan(camFOV/2), -range(2)*tan(camFOV/2)];
ffarlength = abs(fRH(2) - fLH(2));
fnearlength = abs(fRH(1) - fLH(1));
hyp = [norm([fnearlength/2,range(1)]), norm([ffarlength/2,range(2)])];

%Rotate above shape to desired heading/camera angle combination
rfar = norm([fRH(2),flines(2)]);  %projected radial distance to far pt
rnear = norm([fLH(1),flines(1)]); %projected radial distance to near pt
ranglefar = atan2(fRH(2),flines(2));
ranglenear = atan2(fRH(1),flines(1));
ranglefar2 = [-ranglefar+camAngle+hdg_u, ranglefar+camAngle+hdg_u];
ranglenear2 = [-ranglenear+camAngle+hdg_u, ranglenear+camAngle+hdg_u];
FOV1 = [x_u+rfar*cos(ranglefar2(2)),y_u+rfar*sin(ranglefar2(2))];
%top RH
FOV2 = [x_u+rfar*cos(ranglefar2(1)), y_u+rfar*sin(ranglefar2(1))];
%top LH
FOV3 = [x_u+rnear*cos(ranglenear2(2)), y_u+rnear*sin(ranglenear2(2))];
%bottom LH
FOV4 = [x_u+rnear*cos(ranglenear2(1)), y_u+rnear*sin(ranglenear2(1))];
%bottom RH
FOVPLOT=[FOV1;FOV2;FOV4;FOV3;FOV1];
if plotflag==1,
    %plot(FOVPLOT(:,1),FOVPLOT(:,2), 'r'),
end

return  %createFOV
```

```
function [x_u2,y_u2,x_us,y_us,hdg_uw,ttc,tti,x_t_e,y_t_e] =
turnpath(mtr,hdg_uw,Wx,Wy,hdg_u,vel_u,x_u,y_u,hdg_t,x_t,y_t,vel_t,camAn
gle,d,turnside,plotflag);
%turnpath is responsible for calculating

d2r = pi/180;
sw=0;
% turnside = 'rh';
hdg_uwprev = hdg_uw;
for tangentloop = 1:5 %This will close in on the final turn-to-course
exit angle
    %The following is a series of if-then statements that deal with
    %peculiarities of the target being in different quadrants
    if hdg_u>pi & hdg_u < 2*pi
        if hdg_uw>0
            if turnside=='rh'
                chi = hdg_u - hdg_uw(1);
            else %'lh'
              chi =  2*pi - hdg_u + hdg_uw(1); %determine amount a/c is
off intended course
            end
        else %hdg_uw<0
            if turnside=='rh',
                chi = hdg_u - hdg_uw(1);
            else %'lh'
              chi =  2*pi - hdg_u + hdg_uw(1); %determine amount a/c is
off intended course
            end
        end
    else % 0<hdg_u<pi
        if hdg_uw<0
            if turnside == 'lh',
                chi = (2*pi+hdg_uw(1)) - hdg_u;
            else
                chi =  hdg_u - hdg_uw(1);
            end
        else
            chi =  hdg_u - hdg_uw(1); %determine amount a/c is off
intended course
        end
    end
    %other special case
    ttc = abs(chi/mtr); %time-to-course: time it takes to turn to new
course
    arclength = vel_u*ttc*sign(chi);%scribe arc
    R = arclength/chi; %negative implies turn to LHS of uav
    if turnside=='lh',R = -abs(R);else R = abs(R); end
    if arclength == 0, R=0,end
    %determine center for uav turn radius: 90deg off heading (r or l)
    Rcenter = [x_u+R*sin(hdg_u), y_u-R*cos(hdg_u)];
    [xCircle yCircle] = ellipse(R,R,0,Rcenter(1),Rcenter(2),100);
    %     plot(Rcenter(1),Rcenter(2),'.');
    %     plot(xCircle,yCircle, 'b')
    %Commence Turn
    P = abs(round(chi/d2r/3)); hdg_uu = hdg_u; x_u2=x_u; y_u2=y_u;
```

```
    for j = 1:P %create a loop to turn UAV to req'd heading
        hdg_uu = hdg_uu + mtr*ttc/P*sign(-R);
        x_u2 = x_u2 + vel_u*cos(hdg_uu)*ttc/P - Wx*ttc/P;
        y_u2 = y_u2 + vel_u*sin(hdg_uu)*ttc/P - Wy*ttc/P;
        [x_s,y_s] = getSensorLoc(x_u2, y_u2, hdg_uu, camAngle,
d,plotflag,'sensor');
        if plotflag==1,
            %plot(x_u2,y_u2,'bo');
        end
    end
    hdg_uwprev = hdg_uw; %save heading calc from this run
    x_t2 = x_t + vel_t*(cos(hdg_t))*ttc; %expected pos of target
    y_t2 = y_t + vel_t*(sin(hdg_t))*ttc;
    [x_us y_us x_t_e y_t_e hdg_uw, tti count] = CourseOpt(vel_u, vel_t,
hdg_t, Wx, Wy, x_t2, y_t2, x_u2, y_u2,camAngle,d, hdg_u);

end %tangentloop

return  %turnpath
```

Appendix C. Pathmaker Validation Flight Tests

First Flight – Stationary Target Identification

The goals for this flight are to ensure the UAV is working correctly since the last flight, and then test the ability to track a stationary target. An individual will be sent out to an unspecified location away from the ground control unit, where the UAV will be required to positively identify him upon flight path execution.

I. Preflight

1. Range Checks
    a. Verify packets per second (PPS) are between 20 and 30
    b. Walk the airplane out while maintaining line of sight with the Commbox
    c. Monitor the PPS in the ground station.
    d. Verify PPS between 15 and 30 within 250 m line of sight
2. Sensor Checks
    a. Position the aircraft in level flight attitude
    b. Verify the horizon is approximately level
    Note: If the horizon indicator does not indicate level, select "Attitude Level" button in the Autopilot Variable window
    c. Bank the aircraft *right* 45 degrees and hold attitude for at least 5 seconds. Verify attitude indicator reads correctly
    d. Bank the aircraft *left* 45 degrees and hold attitude for at least 5 seconds. Verify attitude indicator reads correctly
    e. Pitch the aircraft *up* 45 degrees and hold attitude for at least 5 seconds. Verify attitude indicator reads correctly
    f. Pitch the aircraft *down* 45 degrees and hold attitude for at least 5 seconds. Verify attitude indicator reads correctly
    g. Press the "Zero Pressure" button
    h. Verify the indicated airspeed is within 5 kts of zero
    i. Verify the indicated altitude is within 10 ft of zero
    j. Cover the pitot tube and verify that the airspeed increases slightly
    k. Place aircraft on level ground
    l. Press the "Check Sensors" button. Ensure all sensors pass (see message window)
3. Stationary Target Placement
    a. Send an individual (target) out with a GPS to an unspecified location within the test range.
        1. This location should be at least 300 meters from the ground station and not be obvious to the person manning the UAV ground station.
        2. For testing purposes, the person should wear something that highly contrasts the surroundings, i.e. a red jacket, vest, etc.
    b. When the target is satisfied with his location, use the GPS to mark the lat/long.
    c. Relay the lat/long data to the ground station member. Standard convention for relaying target data will be as follows:
        1. Latitude
        2. Longitude

3. Target Course
4. Target Speed
5. Time of change to current course and speed.

d. Have target announce his location in generalities, i.e., "I'm south of you". This will be used to position the UAV away from final target for beginning of test.

II. Launch

4. Launch the aircraft and trim it for level flight using the trim tabs on the RC controller.

III. Flight

Note: The communication quality should be monitored throughout the flight

5. Maintain level flight. Verify the following:
    a. GPS maintains lock
    b. Airspeed and Altitude values in the artificial horizon are reasonable
    c. Roll, Pitch, and Heading angles shown in the artificial horizon
6. Open the servo Setup Screen in Virtual Cockpit
7. Load the Camp Attebury waypoint racetrack into Virtual Cockpit.
8. With Virtual Cockpit set to "Nav" mode, instruct the pilot to hand off the aircraft to Autopilot.
9. If the autopilot appears to be tracking well with no irregularities (follows the waypoints correctly, adjusts for wind, no instabilities, etc) we are ready for target ID.
10. Ensure UAV flight data is being recorded.
11. Using Onpoint, begin recording the video stream.
12. Using noted general location of target, position UAV at least 400m away from general location of target noted above.
13. With the target data loaded into the FTC, execute Pathmaker.
14. Take note of the final waypoint. Prepare to identify the target.
15. Announce when you are able to identify the target. Note if you were able to detect the target in the front camera, side camera, or both.
16. Make note of position target in FOV during loiter. Does it stay in the center of the FOV? Is the turn radius too big or too small?
17. Using OnPoint, try to lock onto the target to receive lat/long data.
18. Stop Recording video stream.

IV. Landing

19. Land the aircraft in RC Mode, ensuring airplane power remains on

V. Postflight

20. If the UAV was unsuccessful in keeping the target in the FOV during the turn, a change in the loiter radius may be required.

Second Flight- tracking a moving target

The second flight will build on the success of the 1st flight. This flight will measure the ability of Pathmaker to adequately predict the future location of a target that is traveling along a specified route.

I. Preflight

Stationary Target Placement

1. Do a time hack so that all watches are in sync.
2. Send an individual (target) out with a GPS to an unspecified location within the test range.
    a. This location should be at least 300 meters from the ground station and not be obvious to the person manning the UAV ground station.
    b. For testing purposes, the person should wear something that sticks out from the surroundings, i.e. a red jacket, vest, etc.
3. When the target is satisfied with his location, Use the GPS to mark the lat/long.
4. Have target announce his location in generalities, i.e., "I'm south of you". This will be used to position the UAV away from final target for beginning of test.
5. The target shall determine the direction he intends to travel.

NOTE: You may want to wait until the UAV is airborne to commence the following steps.

6. The target will mark the time, and commence walking at a constant course and speed in a direction of his choice.
7. Monitor speed and course on the GPS. Once satisfied that the GPS has stabilized on your course and speed, relay the lat/long, target course and speed, and time data to the ground station member. Standard convention for relaying target data will be as follows:
    a. Latitude
    b. Longitude
    c. Target Course
    d. Target Speed
    e. Time of change to current course and speed.
8. The ground station member should input the target information into the FTC

II. Launch

1. Same as above

III. Flight

0. If the autopilot appears to be tracking well with no irregularities (follows the waypoints correctly, adjusts for wind, no instabilities, etc) we are ready for target ID.
1. Ensure UAV flight data is being recorded.
2. Using Onpoint, begin recording the video stream.

3. Using noted general location of target, position UAV at least 400m away from general location of target noted above.
4. With the target data loaded into the FTC, execute Pathmaker.
5. Take note of the final waypoint. Prepare to identify the target.
6. Announce when you are able to identify the target. Note if you were able to detect the target in the front camera, side camera, or both.
7. Make note of position target in FOV during loiter. Does it stay in the center of the FOV? Is the turn radius too big or too small?
8. Using OnPoint, try to lock onto the target to receive lat/long data.
9. Stop Recording video stream.

IV. Landing

16. Land the aircraft in RC Mode, ensuring airplane power remains on

Third Flight – stop and go targets

The third flight will build on the success of the 1st two flights. This flight will measure the ability of Pathmaker to adequately predict the future location of a target with nonlinear trajectory.

I. Preflight

1. Stationary Target Placement
    a. Do a time hack so that all watches are in sync.
    b. Send an individual (target) out with a GPS to an unspecified location within the test range.
        1. This location should be at least 300 meters from the ground station and not be obvious to the person manning the UAV ground station.
        2. For testing purposes, the person should wear something that sticks out from the surroundings, i.e. a red jacket, vest, etc.
    c. When the target is satisfied with his location, Use the GPS to mark the lat/long.
    d. Have target announce his location in generalities, i.e., "I'm south of you". This will be used to position the UAV away from final target for beginning of test.
    e. Determine the intended direction of travel.
    f. The target will mark the time, and commence walking at a constant course and speed in a direction of his choice.
    g. Monitor speed and course on the GPS. Once satisfied that the GPS has stabilized on your course and speed
    h. Relay the lat/long, target course and speed, and time data to the ground station member. Standard convention for relaying target data will be as follows:
        1. Latitude
        2. Longitude
        3. Target Course
        4. Target Speed

5. Time of change to current course and speed.

i. The ground station member should input the target information into the FTC

II. Launch

2. Same as above

III. Flight

3. If the autopilot appears to be tracking well with no irregularities (follows the waypoints correctly, adjusts for wind, no instabilities, etc) we are ready for target ID.
4. Ensure UAV flight data is being recorded.
5. Using Onpoint, begin recording the video stream.
6. Using noted general location of target, position UAV at least 400m away from general location of target noted above.
7. With the target data loaded into the FTC, execute Pathmaker.
8. At this time, the target should do the following:
    i. Stop
    ii. Mark the time he stopped, and note the lat/long of his current location.
    iii. Relay this new target data to the GCU.
9. Upon receiving the updated target data, Load the new data into the FTC and re-execute Pathmaker.
10. The above process of stop and go can be repeated multiple times, but will largely be dependant on the range of the target from UAV.
11. The GCU operator should announce when they are able to identify the target. Note if you were able to detect the target in the front camera, side camera, or both.
12. Upon loitering the target, use OnPoint to determine the lat/long position of the target.
13. Use OnPoint to successfully loiter around moving target. Note effectiveness of Onpoint to maintain track.
14. Make note of position target in FOV during loiter. Does it stay in the center of the FOV? Is the turn radius too big or too small?
15. Stop Recording video stream.

IV. Landing

16. Land the aircraft in RC Mode, ensuring airplane power remains on

Fourth Flight – unpredictable moving target

The fourth flight will build on the success of the 1st three flights. This flight will measure the ability of Pathmaker to adequately predict the future location of a target with nonlinear trajectory. This procedure will have the greatest potential for course prediction errors out of the four.

.

I. Preflight

1. Stationary Target Placement
    j. Do a time hack so that all watches are in sync.
    k. Send an individual (target) out with a GPS to an unspecified location within the test range.
        1. This location should be at least 300 meters from the ground station and not be obvious to the person manning the UAV ground station.
        2. For testing purposes, the person should wear something that sticks out from the surroundings, i.e. a red jacket, vest, etc.
    l. When the target is satisfied with his location, Use the GPS to mark the lat/long.
    m. Have target announce his location in generalities, i.e., "I'm south of you". This will be used to position the UAV away from final target for beginning of test.
    n. Determine the intended direction of travel.
    o. The target will mark the time, and commence walking at a constant course and speed in a direction of his choice.
    p. Monitor speed and course on the GPS. Once satisfied that the GPS has stabilized on your course and speed
    q. Relay the lat/long, target course and speed, and time data to the ground station member. Standard convention for relaying target data will be as follows:
        1. Latitude
        2. Longitude
        3. Target Course
        4. Target Speed
        5. Time of change to current course and speed.
    r. The ground station member should input the target information into the FTC

II. Launch

2. Same as above

III. Flight

3. If the autopilot appears to be tracking well with no irregularities (follows the waypoints correctly, adjusts for wind, no instabilities, etc) we are ready for target ID.
4. Ensure UAV flight data is being recorded.
5. Using Onpoint, begin recording the video stream.

6. Using noted general location of target, position UAV at least 400m away from general location of target noted above.
7. With the target data loaded into the FTC, execute Pathmaker.
8. At this time, the target should do the following:
    iv. Change course, preferably by some angle greater than 90 degrees.
    v. Mark the time he stopped, and note the lat/long of his current location.
    vi. Relay this new target data to the GCU.
9. Upon receiving the updated target data, Load the new data into the FTC and re-execute Pathmaker.
10. The above process of stop and go can be repeated multiple times, but will largely be dependant on the range of the target from UAV.
11. The GCU operator should announce when they are able to identify the target. Note if you were able to detect the target in the front camera, side camera, or both.
12. Upon loitering the target, use OnPoint to determine the lat/long position of the target.
13. Use OnPoint to successfully loiter around moving target. Note effectiveness of Onpoint to maintain track.
14. Make note of position target in FOV during loiter. Does it stay in the center of the FOV? Is the turn radius too big or too small?
15. Stop Recording video stream.

IV. Landing

16. Land the aircraft in RC Mode, ensuring airplane power remains on

Appendix D. Hardware Setup

Overview

A key aspect of the current thesis was the actual implementation of optimal flight path solutions into a real-world system. The following chapter will try to document the various hardware and software systems used within the thesis work, and act as a reference guide for future work.

Kestrel Autopilot Setup

The Kestrel Autopilot is an integral part of system. The following section will step through the setup of the Kestrel Autopilot in a fashion that is tailored specifically for the current thesis application. All work should also be referenced to Kestrel's user manual.

Setting up comms with laptop

This section provides a step-by-step instruction for setting up communications between the laptop, Commbox, and autopilot. This setup is required for talking to the autopilot and making any changes to Flash files, PID settings, or any autopilot settings in general. It will allow you to interface with the autopilot and force servo commands, but it will not allow any sort of Hardware-in-the-Loop (HIL) ability.

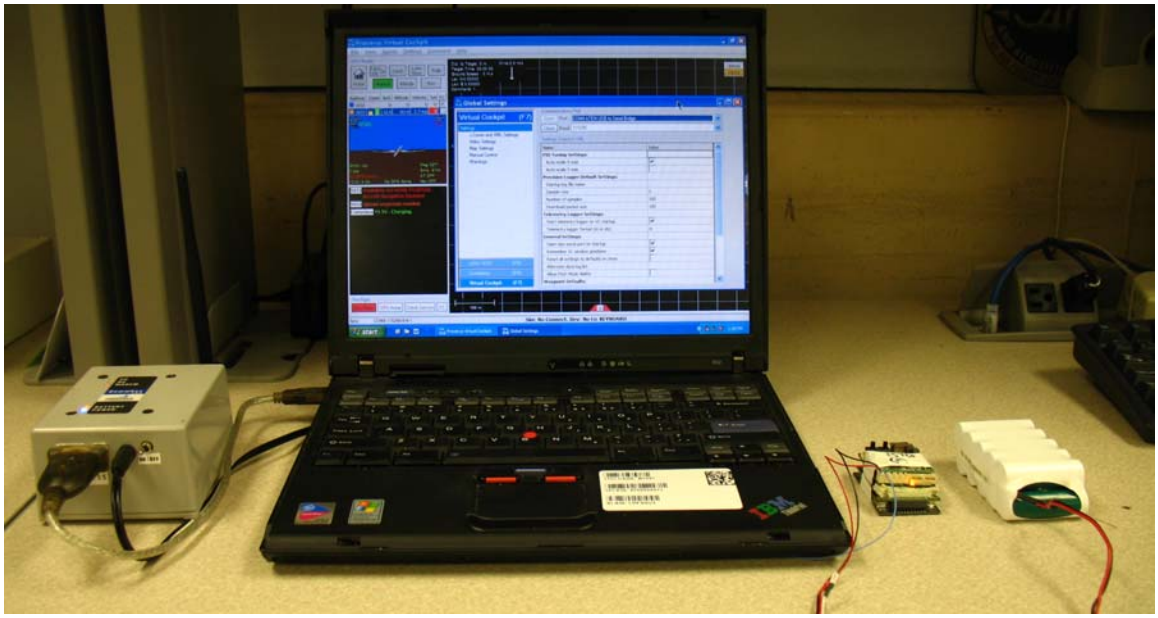


Figure 32. Representative comms setup

1. Connect battery to autopilot.
   a. You should get a steady orange light, a blinking red light, and a steady green light.
2. Turn on Commbox.
   a. With the commbox plugged into a power outlet, you should have a blue power light, orange charging battery light, and red blinking modem light.
3. Using a Serial to USB connector cord, plug the Commbox into one of the available USB ports. In this example we will use the top USB port on the current laptop setup.
4. Open Virtual Cockpit
5. Go to "Settings", "Comm and XML" (see Figure 33)
6. On Communications Port page, set the port to the correct com port. On the current laptop, the top USB port on the LH side of the laptop will show up as

"COM 4 AETN USB to Serial Bridge". If for some reason this COM port does not work, try the others. Virtual Cockpit is known to be finicky in this regard.

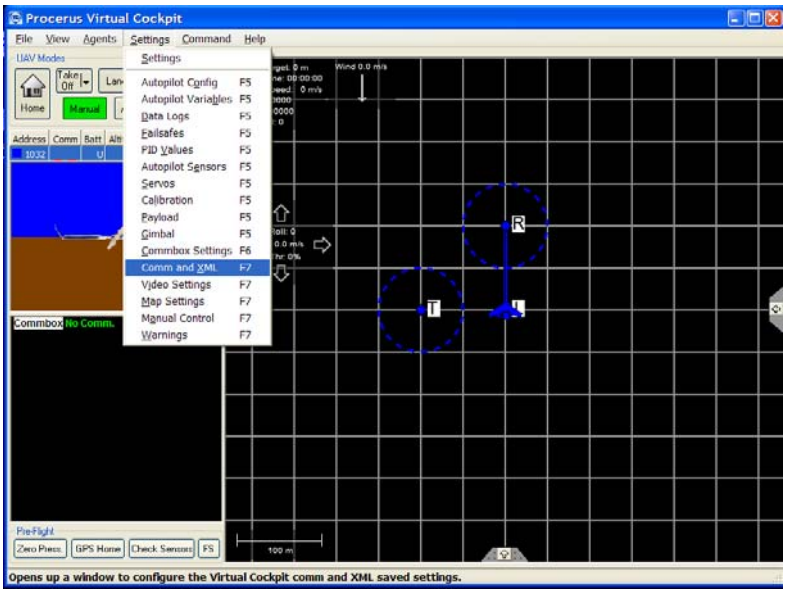


Figure 33. Locating "Comm and XML" from pull down menu

7. Do not change the baud rate or any other settings on this page at this time.
8. With the correct com port selected, you should now see two yellow or green power bars within the HIS screen of virtual cockpit. The Commbox voltage should also appear in comment box area. You may also see a 5 second count-down when the autopilot is initially hooked up.
   a. The Commbox should now have blinking orange VC light as well as the blinking Modem light.
   b. The green light on the autopilot should now be blinking, rather than steady.

9. You are now set up sufficiently to communicate with the autopilot. This configuration is sufficient for uploading or downloading PID and Flash values, as well as accessing any other information on the autopilot.
10. This would also be the general startup procedure used for initialization of the Autopilot system when used in actual flight test scenario as well.

The next section will go through a normal Hardware-In-the-Loop setup.

Hardware-In-the Loop (HIL) setup

While the above setup is good for basic communication with the autopilot, often times you will want to test certain scenarios out using a HIL setup. This section will describe the hardware and process necessary to perform a basic HIL setup with the Kestrel Autopilot.

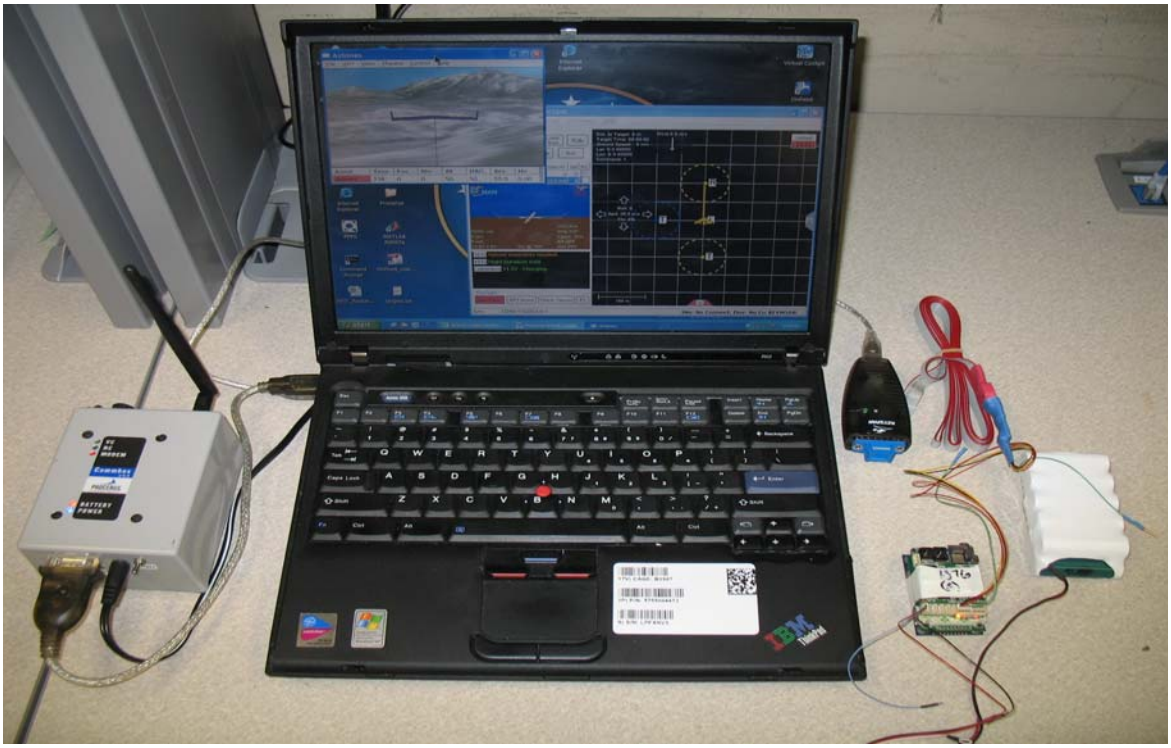


Figure 34. Representative HIL setup

1. Maintain the same setup as in section 4.4.1; however remove power from the autopilot and commbox.
2. Obtain the comm cable shown below, and plug into computer using a serial-to-USB converter. In the following demonstration it will be plugged into the lower USB port on the laptop computer.

a. Note that the green wire must be **unconnected.** If the green wire is connected, the serial cable will not communicate with the autopilot.

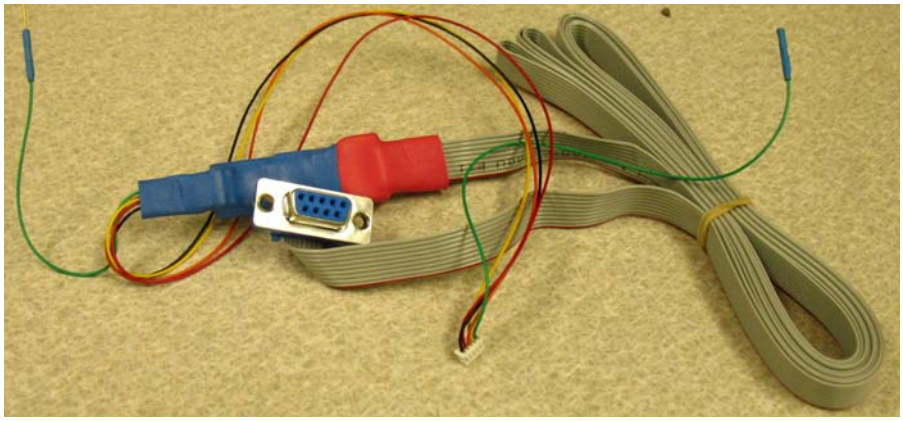


Figure 35. HIL Comm Cable

3. Plug the autopilot comm serial wire into the "serial A" slot. See Figure 36

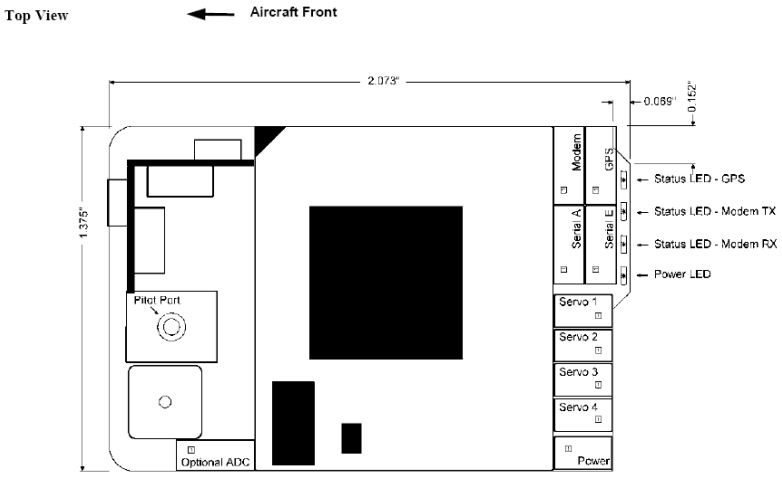


Figure 36. Diagram of Autopilot Ports

4. Power up both the autopilot and Commbox.

5. Open Virtual Cockpit.

6. Click on "Settings", "Payload"

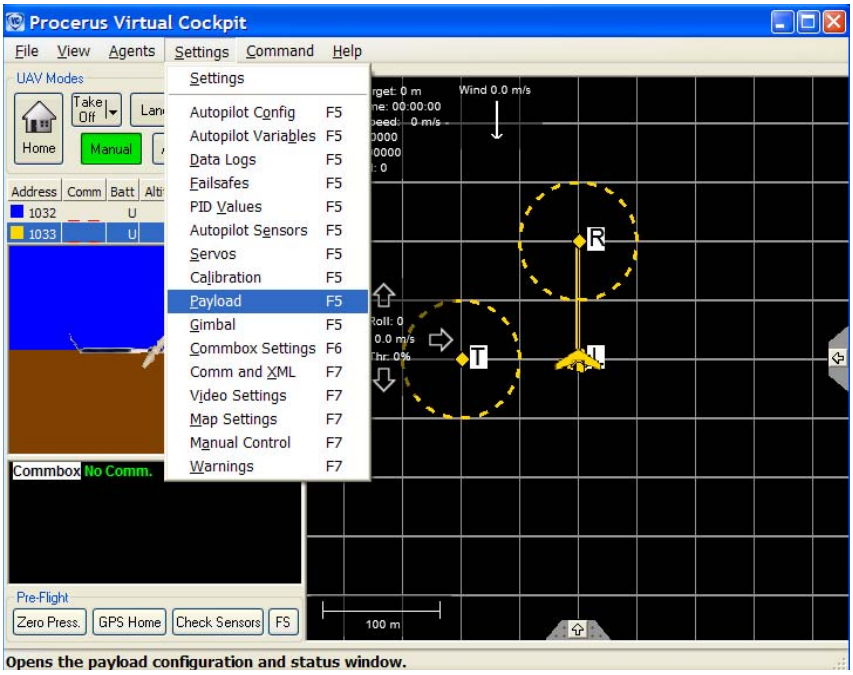


Figure 37. Locating "Payload" from pulldown menu

7. From the Payload page, for "Port A", select "Pins 3,4 for Hardware-In-The-Loop".

   a. Port E can be any of the options at this time and it will not affect the HIL setup. Port E is used primarily for setting up a gimbaled camera.

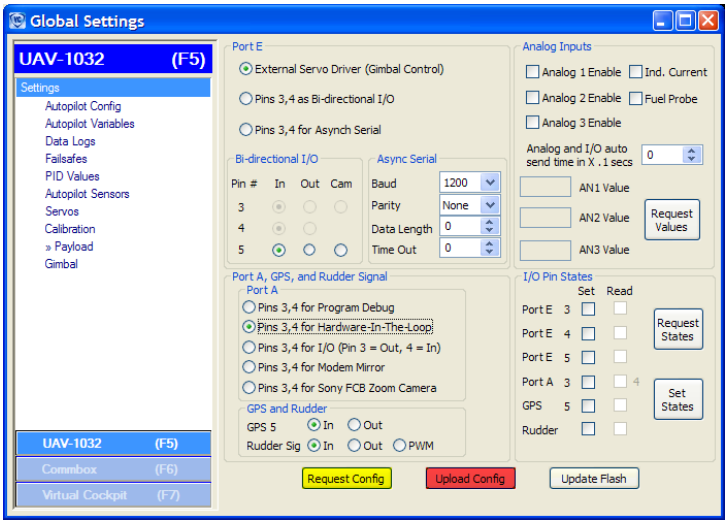


Figure 38. Payload Window

8. Hit the “upload Config” button, followed by the “Update Flash” button. This will have loaded and saved the new settings to the autopilot.
9. Unplug the power for the Autopilot, and plug it back in after all the autopilot lights have extinguished. This resets the autopilot.
10. Open the Aviones.
11. Click on “UAV”, “Add new…”

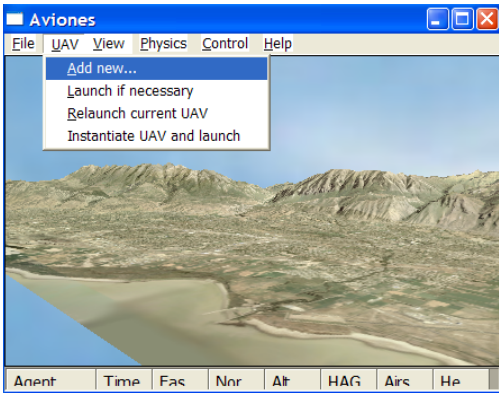


Figure 39. Locating “Add new…” from pulldown menu

a. If an “Add UAV” pop-up screen does not immediately appear, hit the Alt button on the keyboard.

12. In the “Add UAV” pop-up, hit OK.
13. Click on “View”, “HIL Sim Control”
14. In the HIL Simulation Control pop-up, you will need to enter the correct “Serial Port #”. Common serial port numbers are 5 for the lower USB port.
15. With the correct port number selected, you should see a count-down in the main virtual cockpit window, and then another window will pop up requesting launch

altitude. This is the altitude you are assuming the UAV you see in the Aviones screen is starting at. Select any of the three choices you prefer.

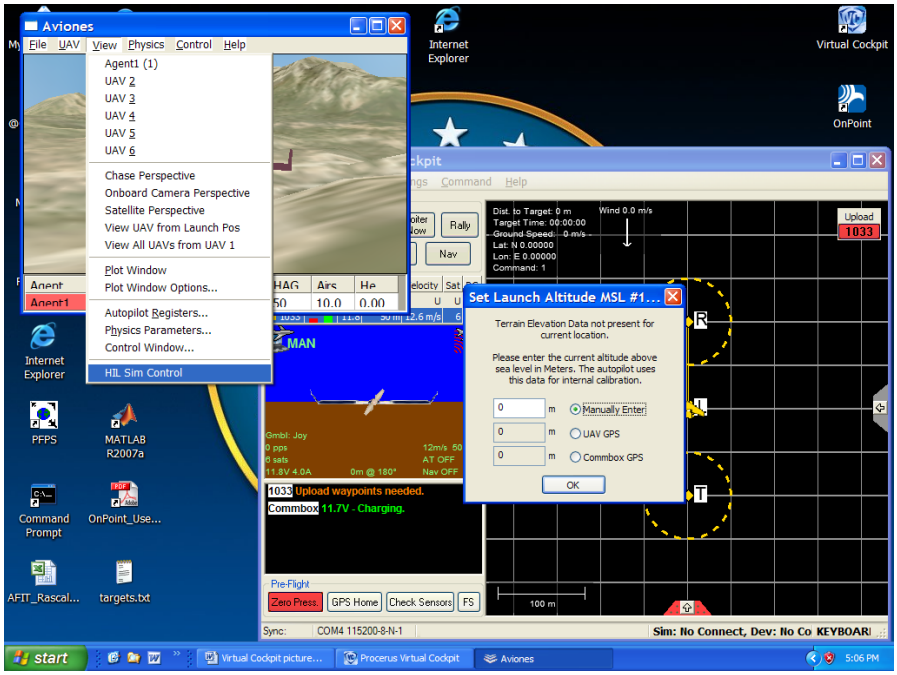


Figure 40. Locating "HIL Sim Control" from pulldown menu

16. You should now have a solid orange light, and a blinking red, green and red light on the autopilot. The outside right light indicates that the HIL setup is communicating with Virtual Cockpit. On the Commbox you should have a blinking red modem light and blinking orange VC light, as well as illuminated battery and power lights.

17. At this point you are ready to conduct a HIL flight. To launch the aircraft, click on "UAV" and "Launch if necessary". The autopilot will now begin to fly via commands sent from Virtual Cockpit. Follow standard operating procedures for setting up waypoints as described in the Procerus instruction manual.

Setting physics.dll parameters

It is necessary to modify the physics.dll file to match your current aircraft as best as possible. A Sig Rascal model was created using empirical data from Jodeh's (Jodeh, 2006) thesis. Units of measure are either in standard metric units, or follow the units of measure prescribed in Appendix B of Nelson's "Flight Stability and Control" (Nelson, 1998: 401)

1. The easiest way to alter the physics.dll parameters is to import them into Excel.
2. Locate the "physics_params.txt" file, which should be located in the parent Aviones directory.
3. Using Excel, open the "physics_params.txt" file as a comma delimited file in Excel.
4. Modify as needed.

External Servo Board Setup

1. Follow the updated instructions given in section 1.5.4 of Procerus' Installation Configuration guide V2.3 (Kestrel Autopilot System, 2007:19).
2. It should be noted that the external servo board needs a power source external to the Serial E connector.

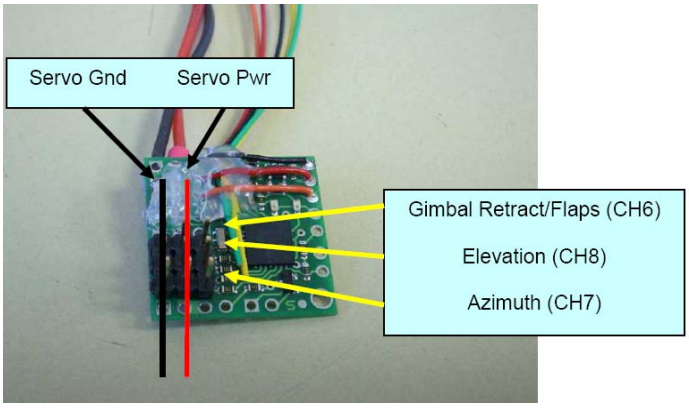


Figure 41. External Servo Board

3. The channels on the servo expansion board are very specific, and cannot be reprogrammed.
4. To control channels 7 and 8, you will need to use the manual controller setup. From Virtual Cockpit, go to "Settings", "Manual Control".
5. In the "Joystick Mapping" Section, select "User Configured".

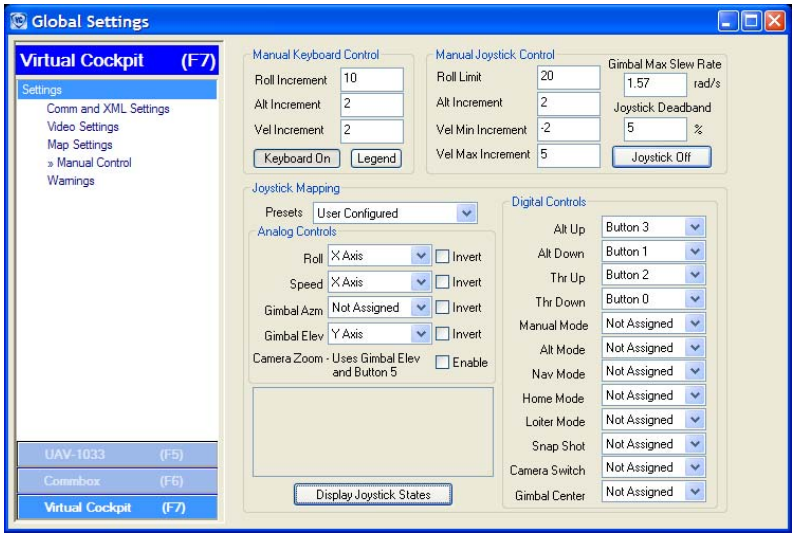


Figure 42. Payload Window

6. Using an available USB gamepad, assign gimbal azimuth and elevation axis relating to the controller you are using. If you are not sure which axis/button is relates to which, click on the "Display Joystick States". Afterward, when you click on a gamepad button or move one of the controllers, it will display the pertinent button or axis number.

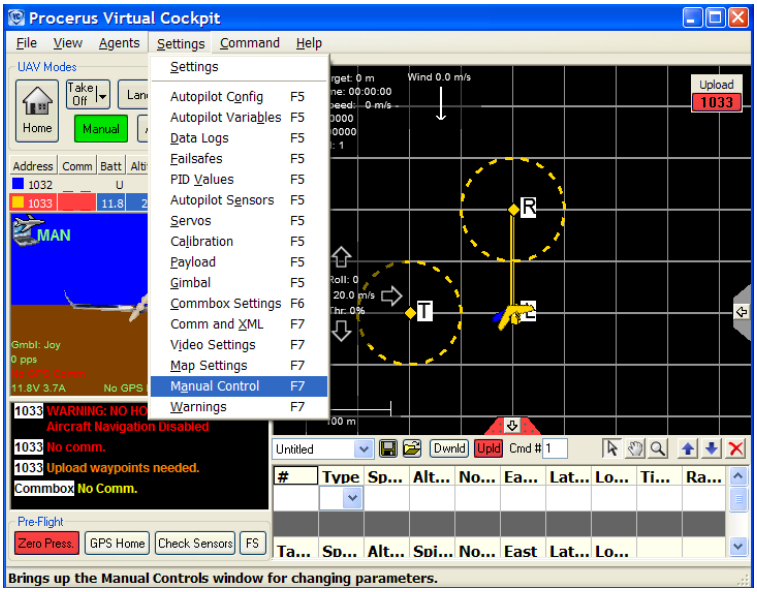


Figure 43. Locating "Manual Control" from pull down menu

RC controller setup

Some items should be noted for setup of the RC controller.

1. The RC controller should be set to “PPM”
    a. If “PPM” is not selected, do the following:
    b. On the Futaba front face, press and hold the “Mode/Page” button.

Figure 44. RC controller display and interface

    c. Using the selector wheel, scroll down to page 2 and select “PARAMETER” by pressing the selector wheel.
    d. Using the Select (Cursor) buttons, select “MODUL” .
    e. Change “MODUL” setting to “PPM” using the selector wheel.

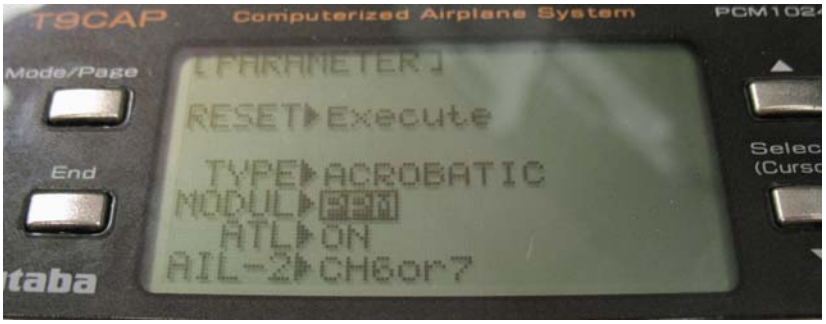


Figure 45. Locating "PPM" from controller menu

f. Press the "End" button twice to exit back to the main menu.

2. Other parameter may be changed on the RC controller box in a similar fashion as described above.

Virtual Cockpit Setup

Setting Flash Values

1. Setup comms with laptop as in section 4.1.1

2. From Virtual Cockpit, click on “Settings,” “Autopilot Variables”

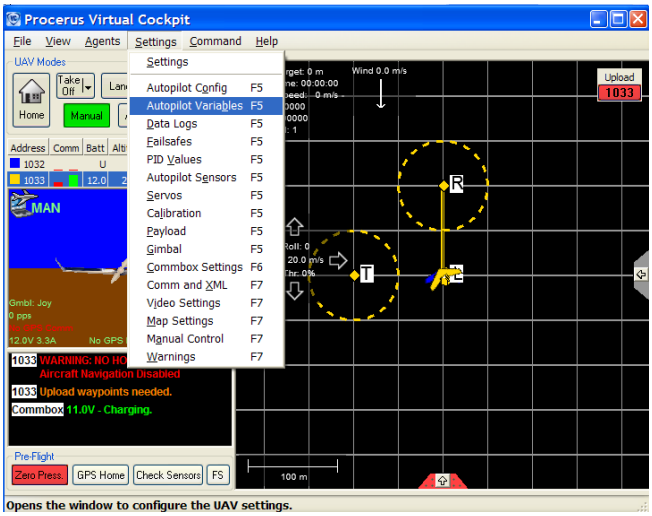


Figure 46. Locating “Autopilot Variables” from pull down menu

3. You are now at a page that can be used to change the PID values

Setting PID values

4. Setup comms with laptop as in section 4.1.1

5. From Virtual Cockpit, click on “Settings,” “PID values”

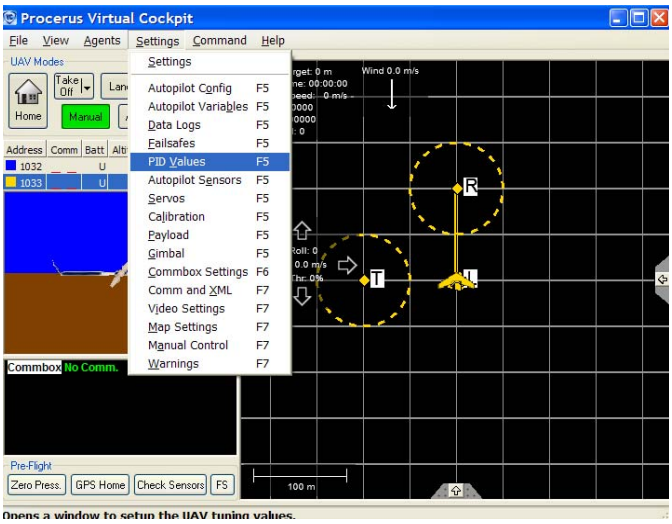


Figure 47. Locating “PID Values” from pull down menu

6. You are now at a page that can be used to change the PID values

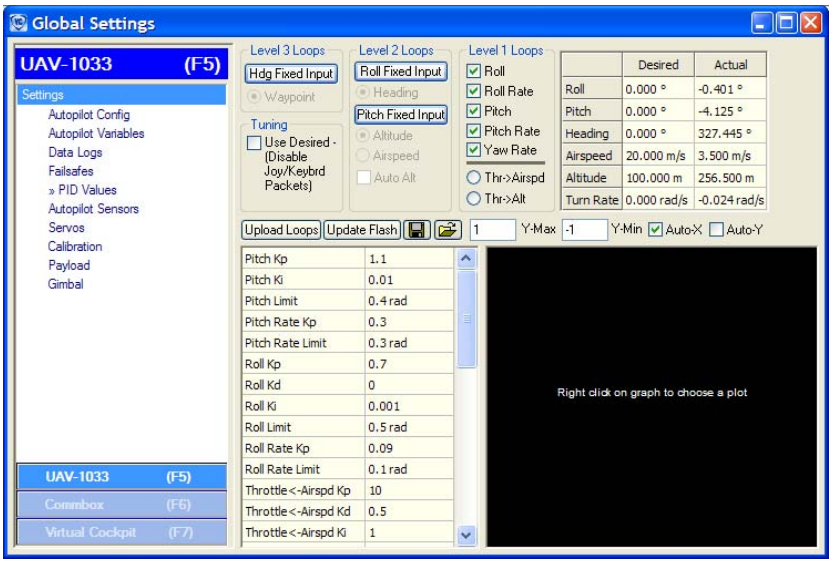


Figure 48. Locating “PID Values” from pull down menu

Bibliography

"Aerocomm Instant Wireless Communications." http://aerocomm.com/rf_transceiver_modules/ac4790_mesh-ready_transceiver.htm. 2 February 2007

"AVDS Simulation Software" http://www.rassimtech.com/Commercial.htm. 21 February 2008

Bryson, A., *Dynamic Optimization.* Addison-Wesley; Menlo Park, CA. 1999.

Crouse, Joshua D. *Development Of Cursor-on-Target Control for Semi-Autonomous Unmanned Aircraft Systems*. MS thesis. AFIT/GAE/ENY/07-J04. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 2007.

Dubins, L., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents." *American Journal of Mathematics,* Vol. 79, No. 3 (Jul., 1957), pp. 497-516.

Frew, E., Xiao, X., Spry, S., McGee, T., Kim, Z., Tisdale, J., Sengupta, R., Hendrick, K.J. "Flight Demonstrations of Self-directed Collaborative Navigation of Small Unmanned Aircraft." *Proceedings of the 2004 IEEE Aerospace Conference*, Big Sky, MT, March 2004.

Geiger, B., Horn, J., DeLullo, A., Long, L., "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming." *AIAA Guidance, Navigation, and Control Conference* , August 2006.

Hull, D., "Conversion of Optimal Control Problems into Parameter Optimization Problems." *Journal of Guidance, Control, and Dynamics,* Vol. 20, No. 1, January-February 1997.

Jodeh, Nidal M. *Development Of Autonomous Unmanned Aerial Vehicle Research Platform: Modeling, Simulating, And Flight Testing*. MS thesis. AFIT/GAE/ENY/06M-18. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2006.

"Kestrel Autopilot System: Installation and Configuration Guide, Document Version 1.7" http://www.procerusuav.com/productsKestrelAutopilot.php. Procerus Technologies , 15 October, 2007.

McGee, T., Spry, S. and Hedrick, K., "Optimal Path Planning in a Constant Wind with a Bounded Turning Rate." *AIAA Guidance, Navigation, and Control Conference and Exhibit*, CA, August 2005.

Nelson, Robert C. *Flight Stability and Automatic Control* (Second Edition). Madison, WI. McGraw-Hill, 1998.

Office of the Secretary of Defense. *Unmanned Aerial Vehicles Roadmap 2005 – 2030.* Washington: HQ DOD, August 2005.

"Procerus Technologies : Kestrel Autopilot." http://procerusuav.com/productsKestrelAutopilot.php. 2 February 2007

Rufa, Justin R. *Development of an Experimental Platform for Testing Autonomous UAV Guidance and Control Algorithms*. MS Thesis, AFIT/GAE/ENY/07M-20. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH. March 2007.

"Tamiya America Item #58268 | RC Mammoth Dump Truck." http://www.tamiyausa.com/product/item.php?product-id=58268. 1 February 2007

"The CMUcam2 Manual." http://www.cs.cmu.edu/~cmucam2/index.html. 2 February 2007

Vaglienti, B., Hoag, R., Niculescu, M. *Piccolo System Users Guide.* Hood River OR. Cloud Cap Technology. 18 April 2005.

Zollars, Michael D. *Optimal Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles*. MS Thesis. AFIT/GAE/ENY/07-M28. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH. March 2007

Vita

Captain Nate Terning was born in Luverne, MN. While spending most of his childhood in Zaire, Africa; he graduated from high school in 1994 from Whitefish, Montana. After a short enlisted stint as a crew chief on MC-130's, Nate was selected for the Scholarship for Outstanding Airmen to ROTC in 1995; he graduated Magna Cum Laude and received a commission from Embry-Riddle Aeronautical University (Prescott) in April of 2000.

Nate's first assignment was within the Seek Eagle stores certification group at Eglin AFB, where he specialized in Flutter analysis and flight testing. In August of 2003, he was transferred to Hill Air Force Base where he joined the T-38 Aero Performance group, and later transferred to the A-10 structural engineering working group. In August of 2006, Capt Terning entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation with a Master of Science in Aeronautical Engineering he will be assigned to the Air Force Research Laboratory's Propulsion Directorate.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED (From — To)
14–03–2008	Master's Thesis	Oct 2007 — Mar 2008

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
REAL-TIME NAVIGATION AND FLIGHT PATH GENERATION FOR TRACKING STOP-AND-GO TARGETS WITH MINIATURE AIR VEHICLES	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Nate A. Terning, Capt, USAF	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	AFIT/GAE/ENY/08-M28
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RY, Dr. Alok Das, 2241 Avionics Circle, WPAFB OH 45433 AFRL/RBC, Dr. Mark Mears, 2130 Eighth St, WPAFB, OH 45433	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research effort focuses on using a heuristic approach to determine the optimal flight path required to put an Unmanned Aircraft System's (UAS) sensor on a moving target in the presence of a constant wind field. This thesis builds on past work using dynamic optimization techniques to calculate minimum time to target. The computationally intensive dynamic optimization routines in their current form take a prohibitive amount of time to calculate and ultimately result in erroneous flight path predictions due to inherent execution time latencies. Therefore an iterative, suboptimal heuristic approach was explored to mitigate excessive calculation times and ultimately yield improved flight path predictions. This report not only explores the heuristic techniques used for flight path calculation, but also includes real world application and flight test results in a Micro Air Vehicle equipped with an autopilot.

15. SUBJECT TERMS

Real-time flight path optimization, Iterative, UAV navigation, tracking moving targets, MAV

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Richard G. Cobb
U	U	U	UU	115	19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636, ext 4559; e-mail: Richard.cobb@afit.edu